P.1877/87

3 1987

informatyka

Czym powinna być INFORMATYKA?

Prof. Robert Kowalski o programowaniu logicznym

Języki asemblerowe

WYDAWNICTWO NOT SIGMA

Cena 150 zł

informatyka

3 1987

Nr 3

Miesięcznik Rok XXII

Marzec 1987

Organ Komitetu
Naukowo-Technicznego NOT
ds. Informatyki

KOLEGIUM REDAKCYJNE:

Dr inż. Wacław ISZKOWSKI, mgr Teresa JABŁOŃSKA (sekretarz redakcji), Władysław KLEPACZ (redaktor naczelny), dr inż. Marek MACHURA, Maria PAWLAK (sekretarz redakcji), dr inż. Wiktor RZECZKOWSKI, mgr inż. Jan RYŻKO, mgr Hanna WŁODARSKA, dr inż. Janusz ZALEWSKI (zastępca redaktora naczelnego)

PRZEWODNICZĄCY
RADY PROGRAMOWEJ:

Prof. dr hab. Juliusz Lech KULIKOWSKI

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 01-517 Warszawa, ul. Mickiewicza 18 m. 17, tel. 39-14-34

Zakł. Graf. „Tamka". Zam. 1040-1300/86. Obj. 4,0 ark. druk. Nakład 7700 egz. K-82.

ISSN 0542-9951, INDEKS 36124

Cena egzemplarza 150 zł
Prenumerata roczna 1800 zł

WYDAWNICTWO NACZELNA ORGANIZACJA TECHNICZNA
CZASOPISM I KSIĄŻEK TECHNICZNYCH
SIGMA
00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE: Strona

Nasze miejsce na mapie 1
Janusz Zalewski

Programowanie logiczne i komputery nowej generacji (1) 2
Robert Kowalski

Przechowywanie wiedzy i niewiedzy w bazach danych (2) 5
Witold Łukaszewicz

Czy następuje zmierzch języków asemblerowych? 8
Piotr Cofta

Sieci komputerowe oparte na mikrokomputerach MERA 60 11
Andrzej Grzywak, Jerzy Kołodziej

Użytkowanie Międzyuczelnianej Sieci Komputerowej 15
Stanisław Góral, Krzysztof Janczewski, Józef Janyszek, Zbigniew Zajączkowski

Pakiet zintegrowany COMPUTEX 19
Grzegorz Trawiński

Orkiestra na pięć fajerek, czyli niezależna ocena pakietu Computex 22
M. M. Zalewska

System programowania grafiki komputerów PSG 2.2 23
Wojciech Mokrzycki

Nowa wersja programu szeregującego wysokiego poziomu dla systemu operacyjnego GEORGE-3 25
Przemysław Rokita

Z KRAJU 27

Systemy mikrokomputerowe w zarządzaniu gospodarką

ZE ŚWIATA 28

IBM — numer 1?
Stan usług w dziedzinie banków danych w Japonii
Tłumaczenie maszynowe

TERMINOLOGIA III okł.

Terminologia lokalnych sieci komputerowych (4)

W NAJBLIŻSZYCH NUMERACH:

- Jan Węglarz ocenia działania systemów komputerowych, omawia przedmiot, metody i kierunki rozwoju
- Ewa Gurbiel, Helena Krupicka, Zdzisław Płoski o próbie normalizacji składni języka Logo
- Marek Pawłowski o mikroprogramowaniu
- Anna Syfert i Andrzej Raznowiecki o funkcjach systemu BIOS dla PC/XT
- Janusz Zalewski porównuje systemy BIOS dla rodziny komputerów IBM PC
- Waldemar Dworakowski omawia informacje dotyczące podziału pamięci, obsługi przerwań i najważniejszych portów w systemach PC/XT
- Maria Meler-Kapcia prezentuje system relacyjnej bazy danych SELKO dla minikomputera Mera 400

BIBLIOTEKA ŚLĄSKA
P. 1877/87

Nasze miejsce na mapie

Minął rok istnienia Redakcji w zmienionym składzie. Jest to okres wystarczająco długi, aby spoglądając wstecz stwierdzić, jakie zmiany zaszły w zawartości **Informatyki** i czy ewoluuje ona w pożądanym kierunku. Jest to też właściwy moment do przedstawienia bieżących planów Redakcji i podjęcia próby sformułowania programu, którego brak nam się zarzuca.

Z obserwacji wiem jednak, że programy działania, których było już wiele w naszym kraju, napotykają trudności realizacyjne i w praktyce niewiele wnoszą — jestem więc przeciwny tworzeniu jeszcze jednego. O wiele ważniejsze wydaje mi się stopniowe wypracowanie właściwego profilu czasopisma.

Moim zdaniem, powinno to być czasopismo o szerokim zakresie tematycznym, mające charakter przeglądowy, zastępujące w pewnym stopniu dostęp do literatury, obrazujące raczej przekrój określonej dziedziny, a nie penetrujące jej głębię. Powinno przekazywać na bieżąco nowości z każdej dziedziny informatyki i donosić, co ciekawego dzieje się w kraju i na świecie. Jasne jest przy tym, że nie chodzi tu, na przykład, o podawanie cen poszczególnych wyrobów czy entuzjastyczne opisywanie coraz nowszych modeli komputerów, lecz o rejstrowanie istotnych faktów o trwałym znaczeniu dla rozwoju informatyki. Powinniśmy, na równi, dostarczać wiadomości popularno-informatyczne i opisywać zastosowania. Reasumując, **Informatyka** powinna szerzyć i pogłębiać wiedzę informatyczną wśród profesjonalistów, nie unikać podejmowania tematyki naukowej i jak najszerzej przedstawiać zastosowania.

Jak to osiągnąć? Nie ulega wątpliwości, że profil czasopisma kształtowany jest przez dział artykułów problemowych. Nowym i ważnym, jak sądzę, elementem obrazu **Informatyki** jest cykl tłumaczeń artykułów autorów zagranicznych o międzynarodowej renomie, drukowanych za ich zgodą. Całkiem niespodziewanie okazało się, że nawet wybitni informatycy chętnie godzą się na opublikowanie swoich artykułów w polskim czasopiśmie, a niektórym — o dziwo, w przeciwieństwie do części rodzimych profesorów — nawet na tym zależy. Wydaje się, że jest to już trwały element nowego profilu **Informatyki**.

Należy się spodziewać, że podobnie będzie z wprowadzonym od połowy ubiegłego roku cyklem artykułów sygnowanych przez PTI. Polskie Towarzystwo Informatyczne organizuje corocznie kilka ogólnopolskich konferencji, których materiały rozchodzą się w bardzo ograniczonym nakładzie. Dlatego sądzę, że bardziej wartościowe referaty warto rozpowszechnić, tym bardziej że PTI ma niekwestionowany autorytet w środowisku informatycznym, a zaproszenie do wygłoszenia referatu na zajęciach Szkoły PTI w pewien sposób przesądza o jego dobrym poziomie.

Inny temat, bardzo aktualny w obecnej dobie burzliwego rozwoju mikroinformatyki — komputery osobiste — postanowiliśmy również włączyć na stałe jako materiał cykliczny. Naturalnie, nie będziemy poświęcać miejsca mikrokomputerom o mocy obliczeniowej poniżej umownej granicy IBM PC, przeznacza się im go bowiem dostatecznie dużo w czasopismach popularnych. Choć polski rynek mikrokomputerów profesjonalnych jest zdominowany przez sprzęt kompatybilny z IBM PC, nie zamykamy dostępu artykułom dotyczącym innych rodzajów mikrokomputerów tej klasy. Po przedstawieniu dość podstawowych zagadnień z tej dziedziny, mających charakter wprowadzenia, będziemy stopniowo przechodzić do omawiania tematów bardziej specjalistycznych.

Mniej więcej w połowie ubiegłego roku postanowiliśmy eksperymentalnie wprowadzić dwa nowe działy tematyczne, obejmujące dwie bardzo ważne i dobrze wyodrębnione dziedziny informatyki: bazy danych i sztuczną inteligencję. Teraz można już powiedzieć, że realizacja tego pomysłu zakończyła się powodzeniem. Okazało się, że w obu tych dziedzinach nie brakuje krajowych autorów, którzy mają coś ciekawego do powiedzenia i przez długi czas na pewno nie grozi nam wyczerpanie tej tematyki.

Czytelnicy od dłuższego czasu zaglądający do **Informatyki** z pewnością zauważyli, że jej najbardziej trwałym, choć formalnie nie wyodrębnionym działem, są języki programowania i systemy operacyjne. Rozpoczynając w 1981 r. od prezentacji Ady, przedstawiliśmy do tej pory — przy zachowaniu ciągłości — większość nowoczesnych i najbardziej popularnych języków i systemów operacyjnych. Obecnie na swoją kolej czekają m. in. Occam, Turing i iRMX.

Dziedziną, która sprawia nam najwięcej kłopotów, a powinna mieć trwałe miejsce w **Informatyce**, jest dydaktyka. Na ten temat publikujemy średnio połowę tego, co byśmy chcieli. Jak sądzimy, przyczyną tego stanu jest brak nowych koncepcji w dydaktyce. W każdym razie, wszystkie propozycje napisania artykułów na ten temat będą wnikliwie rozważone przez Redakcję.

Pewną niespodzianką, znamionującą aktualne tendencje rozwojowe techniki komputerowej, jest systematyczny napływ artykułów na temat sieci komputerowych, systemów wieloprocesorowych, przetwarzania rozproszonego itp., które bardzo chętnie publikujemy. Staramy się również zamieszczać opisy zastosowań, nietypowych konstrukcji czy choćby niewielkich usprawnień dotyczących eksploatowanego sprzętu krajowego, gdyż większość nowych rozwiązań jest przedstawiana na łamach Biuletynu MERA. Pewna grupa artykułów ze sfery przetwarzania danych i zarządzania ma szanse na ukazanie się tylko wtedy, gdy będzie dotyczyła konkretnych realizacji. Opisy samych projektów lub metod nie mają większych szans na publikację, gdyż przy dużym tłoku w dziale artykułowym nie mieszczą się w profilu tematycznym.

Wyłożyłem więc rodzaj creda redakcyjnego a nie programu. Wynika to stąd, że profil i poziom merytoryczny **Informatyki** dopiero się kształtuje, a bardziej zdecydowany program należy starannie wypracować. Jedno jest pewne, że **Informatyka** powinna stać się czasopismem zawodowym, ale jak to zrobić, jeszcze dokładnie nie wiadomo.

Pewną gwarancją, że ewolucja będzie postępować we właściwym kierunku, jest skład Kolegium Redakcyjnego, które stanowi dość dobry przekrój środowiska. Reprezentowane w nim są uczelnie (Instytut Informatyki PW), przemysł (IMM), zastosowania (SCO Cyfronet), a także informatyka „pałacowa" (IPI PAN). Ze względów praktycznych, wynikających z potrzeby częstego kontaktu z Redakcją, w skład Kolegium wchodzą wyłącznie mieszkańcy Warszawy, lecz czytając INFORMATYKĘ łatwo spostrzec, że mamy stałą grupę cenionych współpracowników w takich ośrodkach, jak Gdańsk, Katowice, Kraków, Poznań i Wrocław, choć prawie wyłącznie z uczelni.

Kończąc te uwagi, nie można zapomnieć o dość istotnych sprawach, które mogą mieć decydujący wpływ na odbiór **Informatyki** przez Czytelników. Są to: bardzo niskie honoraria, nieterminowość ukazywania się, mały nakład, zła jakość papieru i druku, brak w drukarni znaków specjalnych stosowanych w informatyce itd. Wielu autorów niechętnie pisze do **Informatyki**, ponieważ zbyt długa jest droga od nadesłania artykułu do jego opublikowania, a jego finalna postać odbiega od wyobrażeń autora. Jest to prawda, lecz Redakcja ma stosunkowo niewielki wpływ na to, aby sytuacja zmieniła się pod tym względem na lepsze. O wiele więcej niż od Redakcji zależy tu od Wydawcy, którym jest WCiKT NOT Sigma. Dlatego wdzięczni będziemy Czytelnikom za wszelkie uwagi krytyczne, opinie i sugestie (formalne i merytoryczne), gdyż pomoże nam to osiągnąć wyższy poziom i lepiej spełnić oczekiwania środowiska.

JANUSZ ZALEWSKI

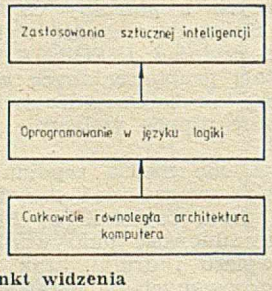
Artykuł ten jest poprawioną wersją wykładu przygotowanego przez Autora na seminarium poświęcone komputerom „piątej generacji", które odbyło się w czasie posiedzenia Stowarzyszenia Europejskich Użytkowników Komputerów Control Data (ECODU), w Londynie, w dniu 21 kwietnia 1986 r. Wcześniejsze wersje artykułu opublikowano również w Future Generation Computing Systems (lipiec 1984) oraz w Datamation (listopad 1984). Tłumaczenie w INFORMATYCE ukazuje się za zgodą Autora.

ROBERT KOWALSKI
Imperial College of Science and Technology
Uniwersytet Londyński, Wielka Brytania

Programowanie logiczne i komputery nowej generacji (I)

W artykule rozważa się programowanie logiczne [1] oraz jego rolę w sztucznej inteligencji i inżynierii oprogramowania, a w szczególności jego potencjalne zastosowania w fazie analizy, w procesie tworzenia programu. Poniżej staram się wykazać, że programowanie logiczne pozwala wykonywać maszynowo analizę systemową (ang. execute); w pewnych wypadkach efektywność wykonania analizy jest na tyle wysoka, że można zupełnie uniknąć oddzielnego tworzenia specyfikacji i budowania programów w konwencjonalnym języku programowania. Jako przykład ogólniejszego zadania polegającego na egzekwowaniu reguł i przepisów, niezależnie od tego, czy posiadają one moc prawną czy też nie, rozważono Brytyjski Akt Narodowości (ang. British Nationality Act), sporo uwagi poświęcono również innym związkom inżynierii oprogramowania i sztucznej inteligencji, a także niektórym skutkom nowej technologii programowania, istotnym z punktu widzenia człowieka.

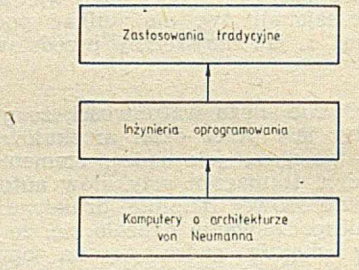
Japończycy jako pierwsi dostrzegli ważność zastosowań sztucznej inteligencji (rys. 1). Uznali oni programowanie logiczne za podstawową technikę programowania i zaproponowali nowe rodzaje architektury komputerów. W gruncie rzeczy te nowe zastosowania są zrozumiałe dla „człowieka z ulicy", zaś architektura komputerów — dla inżyniera elektronika. Ale zanim Japończycy zwrócili uwagę na programowanie logiczne, większość informatyków odrzucała je albo nic o nim nie wiedziała.



Rys. 1. Japoński punkt widzenia

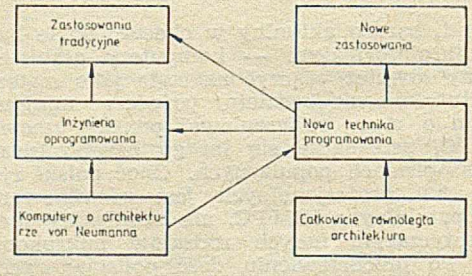
PIĄTA GENERACJA

Punkt widzenia zwykłego informatyka jest tradycyjny, nie tylko co do zastosowań i sprzętu, ale także co do metodologii programowania; konwencjonalne zastosowania naukowe i komercjalne, polegające na „przemiale" danych na komputerach pracujących sekwencyjnie, są programowane krok po kroku i nie ma w nich miejsca na samodzielne podejmowanie przez maszynę jakichkolwiek decyzji strategicznych. Jak więc łączy się konwencjonalne zastosowania i konwencjonalne komputery? Stosuje się konwencjonalne techniki programowania (rys. 2).



Rys. 2. Tradycyjny punkt widzenia

Feigenbaum i McCorduck w książce o komputerach piątej generacji [2] podkreślają ich zasadniczą cechę, jaką jest zastosowanie w dziedzinie systemów ekspertowych. Pomniejszają oni jednak znaczenie nowych architektur i programowania logicznego. Nie chcę tu dowodzić, że Prolog jest lepszy od Lispu, czy że Prolog pasuje do wszystkich dzisiejszych zastosowań. Będę jednak starał się wykazać, że nowa technologia, w połączeniu z programowaniem logicznym, umożliwia nie tylko powstawanie nowych, ale i wspomaganie poprzednich zastosowań (rys. 3).



Rys. 3. Proponowany punkt widzenia

Tak więc należy dokonać rozróżnienia między technologią i zastosowaniami. Nowe technologie programowania, których najbardziej reprezentatywnym przykładem jest programowanie logiczne, nie tylko pozwalają na nowe zastosowania w takich dziedzinach jak systemy ekspertowe czy przetwarzanie języka naturalnego, ale także umożliwiają realizację dotychczasowych zastosowań. Nowe tech-

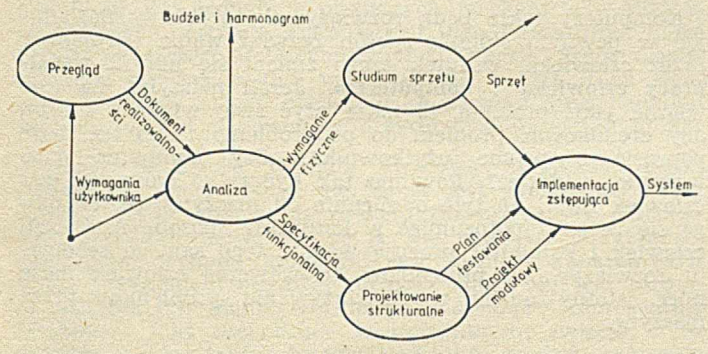
[1] Tytułowy termin „programowanie logiczne" jest tłumaczeniem angielskiego określenia logic programming, które dokładnie oznacza programowanie w języku logiki. Ponieważ pełne określenie jest za długie, a niekiedy spotykana jego skrócona forma — programowanie w logice — jest określeniem żargonowym i nie powinna być używana, redakcja proponuje spopularyzowanie nazwy programowanie logiczne *(przyp. red.)*.

nologie wspomagają różnorodne metodologie programowania, nie tylko dotychczas znane, lecz również nowe metody wytwarzania programów.

Na czym więc polega nowa technologia? Charakteryzuje się ona tym, że pozwala na jawne reprezentowanie wiedzy w formie nie związanej ze sposobami jej użycia do rozwiązywania problemów drogą rozumowania dedukcyjnego, naśladującego rozumowanie człowieka, a więc drogą pokrewną myśleniu człowieka i współpracy człowieka z maszyną. Oznacza to, że nowe oprogramowanie, w porównaniu z konwencjonalnym, pozwala na lepszy wgląd w wiedzę i dzięki temu ułatwia jej rozumienie. Znaczy to także, że można gromadzić wiedzę przyrostowo, gdyż nie jest ona ściśle związana ze sposobami jej wykorzystania. W razie pomyłki lub częstych modyfikacji wiedzy, jak np. w dziedzinie prawa, można ją w łatwy sposób aktualizować.

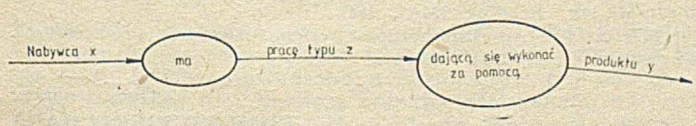
NOWA TECHNOLOGIA A CYKL PRODUKCYJNY OPROGRAMOWANIA

Spójrzmy na miejsce nowej technologii programowania w konwencjonalnym cyklu produkcyjnym oprogramowania (rys. 4 [1]). Przede wszystkim warto zwrócić uwagę na najniżej położoną ścieżkę rysunku związaną z oprogramowaniem. Zaczyna się ona od formułowania wymagań użytkownika, a raczej od tego, co użytkownik sądzi na temat swoich wymagań. W kolejnych fazach analizuje się te wymagania, konstruuje opis funkcjonalny komputerowego rozwiązania problemu, a następnie projektuje oprogramowanie, które jest ostatecznie implementowane metodą zstępującą.



Rys. 4. Cykl rozwoju oprogramowania [1]

Diagram przepływu danych opisujący cykl produkcyjny oprogramowania jest dogodnym narzędziem dla projektanta systemu podczas współpracy z użytkownikiem w fazie analizy. Jest to język, który rozwinęli projektanci, aby łatwiej porozumieć się z ludźmi. Ale diagramy przepływu danych mogą też być interpretowane jako graficzna alternatywa składni dla programowania według reguł. Następującą regułę: **przedsiębiorstwo x jest potencjalnym nabywcą produktu y, jeśli x wykonuje prace typu z, a produkt y pasuje do z**, projektant systemu może wyrazić w postaci procesów i przepływu danych między procesami, obrazując ją graficznie jak na rys. 5.



Rys. 5. Ilustracja reguły: x jest nabywcą produktu y

Postaram się wykazać, że tego rodzaju diagram różni się tylko składnią od języka opartego na logice, uznanego za podstawowy w japońskim projekcie komputerów piątej generacji. Diagram jest równoważny semantycznie językowi reguł złożonych z wniosków i warunków:

x jest potencjalnym nabywcą produktu y
jeżeli x ma pracę typu z
i y pasuje do z.

Podpis pod rysunkiem wyjaśnia cel użycia procesów w diagramie. W tym wypadku celem jest znalezienie produktów poszukiwanych przez klientów. Stanowi to **wniosek reguły**, mianowicie, że pewien produkt y pasuje potencjalnemu klientowi x. Procesy zaznaczone kółkami składają się na **warunki**, które muszą być spełnione, aby wniosek reguły był prawdziwy. Pierwszy proces odnajduje jakiś typ pracy z, którą wykonuje dany klient, a drugi odnajduje produkt y pasujący do z.

Przykład ten ilustruje tezę, że programowanie oparte na logice niekoniecznie musi być programowaniem czy też nawet tworzeniem formalnej specyfikacji. W przykładzie programowanie oparte na logice jest w rzeczywistości **wykonywalną analizą** wymagań użytkownika. Dlatego może ono być wykorzystane w konwencjonalnym cyklu produkcyjnym oprogramowania na bardzo wczesnym etapie. Wymagania użytkownika mogą być przeanalizowane i wykonane zanim uzyska się specyfikację funkcjonalną, projekt lub program. Można wykonać analizę, aby sprawdzić, czy odpowiada ona wymaganiom użytkownika; dzięki temu łatwo wyeliminować nieporozumienia na najwcześniejszym etapie, zanim staną się one przyczyną następnych nieporozumień.

W jaki sposób można wykonywać takie reguły? Prawdopodobnie główną ideą leżącą u podstaw programowania logicznego jest to, że zastosowanie wnioskowania wstecz (ang. backward reasoning) do reguł złożonych z wniosków i warunków zmienia je w procedury. Jeśli, na przykład, dla znanego potencjalnego klienta x należy znaleźć coś, co można mu sprzedać, to procedura otrzymana przez zastosowanie tej reguły wstecz rozbija problem na dwa prostsze podproblemy: zbadaj typ pracy klienta i znajdź produkt, który pasuje do tej pracy. Procedura ta jest zapisana w taki sposób, w jaki wyraża ją człowiek; redukuje ona problemy do podproblemów, aż w końcu nie wymagają one dalszej redukcji. Można ją przedstawić handlowcowi, który nic nie wie o komputerach. Co więcej, niezależnie od tego, czy używa się komputera czy nie, obydwa podproblemy mogą być rozwiązane kolejno przez pojedynczy procesor lub też mogą być rozwiązane równolegle (tak będą pracowały komputery piątej generacji).

W rzeczywistości mamy tu jednak do czynienia z więcej niż jedną procedurą. W pewnym sensie diagram przepływu danych jest mylący dla zasobów wiedzy. Nie jest to bowiem tylko procedura, która dla danego klienta szuka czegoś, co można mu sprzedać. Ta sama wiedza może być użyta do znalezienia klientów na różne produkty. Jeśli trzeba znaleźć klienta x na dany produkt y, to należy dowiedzieć się, jak można przetworzyć dany produkt i znaleźć klienta, który wykonuje właśnie taką pracę. Wiedza może więc być wykorzystana bardziej wszechstronnie niż widział to projektant systemu czy wymagał tego użytkownik. Z punktu widzenia inżynierii oprogramowania problem polega na tym, że użytkownik dysponuje dodatkową wiedzą, której nie uświadamia sobie formułując wymagania.

Analiza strukturalna ma swoje **mocne strony**, a nowa technologia programowania uwydatnia je jeszcze bardziej dzięki możliwości wykonania analizy systemu. Jedną z tych zalet jest fakt, że diagramy przepływu danych są wygodnym, graficznym językiem porozumiewania się z użytkownikami. Są one wystarczająco dokładne, aby projektant systemu mógł wyrazić za ich pomocą wymagania stawiane przez użytkownika — w istocie tak dokładne, że mogą być automatycznie przetłumaczone na reguły, a następnie wykonywane jako procedury.

Diagramy przepływu danych pozwalają też panować nad skalą i złożonością systemu. Reguła DeMarco zaleca, aby ograniczyć diagram przepływu danych do pojedynczej stronicy formatu A4. Gdy zachodzi konieczność wyjścia poza jedną stronicę, przedstawia się dany proces na innej stronicy w postaci bardziej szczegółowych diagramów przepływu danych.

Jakie są **słabe strony** analizy strukturalnej? Użytkownicy nie zawsze jasno uświadamiają sobie swoje potrzeby, a często, gdy zorientują się, o co im chodzi, wycofują się z poprzednich wymagań. Tak więc należy najpierw ustalić, o co chodzi użytkownikom, analizując ich wyobrażenia i wymagania oraz wykorzystując wypowiedzi jako nić prowadzącą do głębiej ukrytej wiedzy. Sądzę, że deklaratywna postać, w jakiej może być wyrażona wiedza przy użyciu techniki programowania logicznego, pomaga uwolnić użytkowników od ich chybionych koncepcji i użyć posiadaną przez nich wiedzę do ambitniejszych celów.

Jakie są więc środki zaradcze zalecane przez takich krytyków analizy strukturalnej jak James Martin [3]? Prawdopodobnie najpopularniejszym jest **szybka budowa prototypów**: wykonuje się prototyp rozwiązania problemu tak szybko i na tak wczesnym etapie, jak to tylko jest możliwe. Jak jednak zaimplementować prototyp? W większości wypadków służy do tego język programowania, przeznaczony do użycia w ostatniej fazie cyklu produkcyjnego oprogramowania. Nowa, oparta na logice, technologia programowania, pozwala implementować prototypy przy użyciu języków przeznaczonych do analizy systemowej, tj. pierwszego etapu cyklu produkcyjnego po wstępnym stadium realizowalności.

Jeśli nie używa się języka programowania, to można skorzystać z generatorów programów czwartej generacji. W wielu wypadkach są one po prostu sparametryzowanymi programami, które użytkownik może samodzielnie przystosować do danego zastosowania, wybierając konkretną kombinację odpowiedzi z ustalonego zestawu możliwości. W innych wypadkach są one oparte na bazach danych.

Coraz częściej relacyjne podejście do baz danych zaczyna ustępować podejściu logicznemu, które jest blisko związane z programowaniem opartym na logice. Można to znowu zilustrować przy użyciu reguły wiążącej potencjalnych klientów z produktami. Regułę tę można uważać za generator zapytań. W zadaniu powiązania klientów z produktami generuje ona zapytanie: **znajdź pewien rodzaj pracy z dla klienta x, do której wykonania pasuje produkt y.**

W terminologii relacyjnych baz danych obydwa warunki związane są relacyjnym operatorem łączenia. Ale nie jest to tylko zapytanie skierowane do konwencjonalnej relacyjnej bazy danych, gdzie cała wiedza jest przechowywana jawnie w postaci tablic, lecz jest to polecenie, którego warunki są obliczane przez inne reguły (lub, co jest równoważne, przez inne procedury). Tak więc, reguły zachowują się tak jak procedury generujące zapytania, na które odpowiada się stosując inne reguły, które z kolei generują następne zapytania.

Wynika z tego, że technologia programowania oparta na logice stanowi wspólną podstawę dla wykonywalnej analizy systemowej i dla baz danych zawierających reguły oraz jawnie przechowywane dane. Ale co nowa technologia programowania ma wspólnego z systemami ekspertowymi?

SYSTEMY EKSPERTOWE

Niżej podany przykład ilustruje, w jakim stopniu nowa technologia nadaje się do zastosowań w dziedzinie systemów ekspertowych. Pochodzi on z logicznej rekonstrukcji systemu ekspertowego, zrealizowanego w EMYCInie przez Petera Alveya w Imperial Cancer Research Institute w Londynie, dokonanej przez Petera Hammonda w Imperial College. Działanie reguły jest takie samo jak w wypadku sprzedawania produktów klientom, choć tym razem mamy do czynienia z leczeniem pacjentów. Reguła wyraża to, że pacjent powinien poddać się kuracji, jeżeli cierpi na dolegliwość, którą łagodzi dana kuracja. Ale w wypadku pacjenta, który przecież jest człowiekiem, należy również zadbać o to, aby kuracja nie wywołała skutków gorszych niż sama choroba. Reguła ma więc w swojej końcowej postaci jeden wniosek i trzy warunki:

x powinien poddać się kuracji y
jeżeli x cierpi na dolegliwość z
i y łagodzi z
i **nie** y jest niewłaściwa dla x.

W tym przykładzie warto zauważyć inną jeszcze cechę języków deklaratywnych: jak łatwo jest modyfikować wiedzę, gdy jest ona zapisana jawnie. Przypuśćmy, że pierwsze sformułowanie reguły zawierało tylko dwa początkowe warunki i wobec tego obchodzono się z ludźmi w taki sam sposób jak z przedsiębiorstwami. Jeśli po jakimś czasie dochodzi się do wniosku, że reguła ma zbyt szeroki zakres, to można go ograniczyć przez podanie dodatkowych warunków. Zmiany takie mogą być zrozumiane i wyjaśnione w terminach deklaratywnych, jasnych dla człowieka, bez potrzeby rozważania ich wpływu na zachowanie komputera.

Należy też zwrócić uwagę na to, że rozważa się tutaj jedynie regułę najwyższego poziomu, mającą na celu powiązanie pacjentów z kuracjami. Można przeanalizować warunki reguły: **co to znaczy, że y łagodzi z; co to znaczy, że y nie jest niewłaściwa dla x.** Można rozwinąć warunki metodą zstępującą w taki sam sposób, w jaki w analizie strukturalnej rozwija się diagram przepływu danych. Zakłada się, że zanegowany warunek „**nie** y jest niewłaściwa dla x" zachodzi wtedy i tylko wtedy, gdy prawdziwość niezanegowanego warunku „y jest niewłaściwa dla x" nie może być dowiedziona. Taki sposób traktowania negacji jest znany pod nazwą „negacja jako porażka" (ang. negation as failure):

y jest niewłaściwa dla x
jeżeli y wzmaga dolegliwość v występującą u x
aspiryna łagodzi zapalenie:
aspiryna łagodzi ból;
...
aspiryna wzmaga rozwój wrzodów trawiennych u x;
alkohol wzmaga wysokie ciśnienie krwi u x;
jeżeli x ma ponad 40 lat
i x jest otyły.

Przykład ten wykazuje, że nowa technologia programowania może być użyta zarówno do realizacji nowych zastosowań, jakimi są systemy ekspertowe, jak też do komputerowego wspomagania konwencjonalnego cyklu tworzenia oprogramowania. Można jednak osiągnąć coś więcej, a mianowicie zmienić naturę samego procesu obliczeniowego i służącą mu metologię programowania. Można skłonić komputery do rozumienia wiedzy wyrażonej w terminach, w jakich zwykle wyraża ją człowiek, i do używania tej wiedzy elastycznie, na różne sposoby i w różnych celach. Można skłonić komputery do rozumienia logiki bezpośrednio jako ich języka wrodzonego.

Komputery takie będą rozwiązywały problemy naśladując w pewnym stopniu sposób rozwiązywania problemów przez człowieka, wskutek czego zmieni się natura **współpracy człowieka z komputerem.** Jeżeli maszyna ma rozwiązać jakiś problem, to może użyć swej własnej wiedzy, aby zredukować problem do podproblemów, lub też może poprosić o pomoc użytkownika. Czy komputer musi wszystko wiedzieć? Powinno tak być tylko wówczas, gdy człowiek myśli na tyle odmiennie od maszyny, że nie mogą się ze sobą porozumieć w naturalny sposób. Ale jeżeli człowiek i komputer pracują według tego samego schematu rozwiązywania problemów, to wówczas człowiek może mieć istotny udział w rozwiązywaniu problemów przez maszynę.

Komputer może **wyjaśnić** swe wnioski prezentując reguły, które do nich doprowadziły. Można jego wyjaśnienia przyjąć lub odrzucić, a można ich użyć do wyciągnięcia zupełnie innych wniosków. Typową cechą podejmowania decyzji przez człowieka jest pytanie innych o radę w celu zrozumienia argumentacji stojącej za wyciągniętymi wnioskami. Po zapoznaniu się z argumentacją należy rozstrzygnąć, czy można się z nią zgodzić czy nie, czy można zaakceptować założenia, które uzasadniają wniosek. Pozwala to panować nad procesem rozwiązywania zadania.

LITERATURA

[1] DeMarco T.: Structured Analysis and System Specification. Yourdon Inc., New York, 1978
[2] Feigenbaum E. A., McCordunck P.: The Fifth Generation. Addison-Wesley. Reading (MA), 1983
[3] Martin J.: An Information System Mefisto. Prentice-Hall, Englewood Cliffs (NJ), 1984.

Tłum. i oprac.:
EDMUND PIERZCHAŁA

Ogłoszenia • Ogłoszenia • Ogłoszenia • Ogłoszenia

BIURO USŁUG KOMPUTEROWYCH. Pośrednictwo sprzedaży mikrokomputerów, części zamiennych. Warszawa, tel. 41-44-48.

EO/272/K/86

Ogłoszenia • Ogłoszenia • Ogłoszenia • Ogłoszenia

WITOLD ŁUKASZEWICZ
Instytut Informatyki
Uniwersytet Warszawski

POLSKIE TOWARZYSTWO INFORMATYCZNE

Przechowywanie wiedzy i niewiedzy w bazach danych (2)

W pierwszej części artykułu omówiono zagadnienia ogólne związane z przechowywaniem wiedzy i niewiedzy w bazach danych w postaci formuł systemu logicznego. W szczególności uzasadniono konieczność stosowania wnioskowania niemonotonicznego w systemach sztucznej inteligencji oraz ukazano niektóre problemy związane z jego formalizacją. Poniżej przedstawiono dwie bardzo popularne formalne metody wnioskowania niemonotonicznego oraz zasygnalizowano inne formalizmy niemonotonicze.

HIPOTEZA DOMKNIĘTOŚCI ŚWIATA

Hipoteza domkniętości świata (ang. closed world assumption) jest jednym z pierwszych mechanizmów niemonotonicznych. Istnieje kilka wersji tego formalizmu, które różnią się szczegółami technicznymi. Wersja, którą przedstawiono poniżej, została wprowadzona w [27]. Inna wersja, znana pod nazwą **negacja jako porażka** (ang. negation as failure), jest używana w programowaniu logicznym [4].

Niech **A** będzie formułą atomową klasycznej logiki pierwszego rzędu. Mówi się, że **A jest zamknięta**, wtedy i tylko wtedy, gdy **A** nie zawiera zmiennych. W przeciwnym wypadku mówi się, że **A jest otwarta.** Przykładem zamkniętej formuły atomowej jest formuła **kanarek (Sylwia)**. Przykładem otwartej formuły atomowej jest formuła **kanarek (x)**. Zamknięta formuła atomowa reprezentuje konkretny fakt zachodzący w opisywanym świecie, a otwarta — zbiór takich faktów. Zbiór ten zawiera wszystkie fakty, które można otrzymać wstawiając w miejsca zmiennych konkretne obiekty.

Hipotezę domkniętości świata otrzymuje się przez dodanie do klasycznej logiki pierwszego rzędu następującej reguły wnioskowania:

jeśli na podstawie posiadanej informacji nie można dowieść, że zamknięta formuła atomowa jest prawdziwa, to należy założyć, że prawdziwa jest jej negacja.

Zastosowanie hipotezy domkniętości świata ilustruje następujący przykład. Rozważmy następującą bazę wiedzy [1]):

(1) pracuje (Jan)
(2) nauczyciel (Piotr)
(3) $\forall$ x pracuje (x) $\supset$ zdrowy (x))

Rozważmy zamkniętą formułę atomową **zdrowy (Piotr)**. Przy użyciu klasycznej logiki pierwszego rzędu formuła ta nie może być wyprowadzona z aksjomatów (1)—(3). Stosując zatem do niej hipotezę domkniętości świata, natychmiast wnioskujemy formułę **$\sim$zdrowy Piotr.** Zauważmy, że hipotezy domkniętości świata nie można zastosować do formuły **zdrowy (Jan)**. Wynika to z faktu, że formułę tę można wyprowadzić z aksjomatów (1)—(3).

Hipoteza domkniętości świata jest za silna, aby uznać ją za właściwe narzędzie do modelowania wnioskowania naturalnego. Widać to wyraźnie na powyższym przykładzie. Wnioskowanie, że osoba jest chora tylko dlatego, że nie można dowieść, iż jest inaczej, jest nie do przyjęcia.

Hipotezę domkniętości świata stosuje się w bazach wiedzy opisujących dziedziny, o których nasza wiedza jest zupełna. Można wówczas pozwolić sobie na usunięcie z bazy wszystkich informacji negatywnych, tzn. negacji zamkniętych formuł atomowych, które i tak będzie można wyprowadzić z hipotezy domkniętości świata. Kiedy nasza wiedza o świecie jest zupełna, informacja negatywna stanowi na ogół ogromny procent całości informacji i wtedy uzyskuje się znaczne ograniczenie zapotrzebowania na pamięć. Ponadto zupełność naszej wiedzy gwarantuje, że hipoteza domkniętości świata nie będzie prowadzić do nieintuicyjnych wniosków. Wszystko bowiem, co będzie można przy jej użyciu wywnioskować, to usunięte informacje negatywne.

Niekontrolowane użycie hipotezy domkniętości świata może prowadzić do sprzeczności. Rozważmy następującą bazę wiedzy:

(1) student (Jan)
(2) uczy-się-angielskiego (Jan) $\supset$ uczy-się-niemieckiego (Jan)

Zauważmy, że aksjomaty **(1)—(2)** nie pozwalają wyprowadzić ani formuły **uczy-się-angielskiego (Jan)**, ani też formuły **uczy-się-niemieckiego (Jan)**. Stosując zatem hipotezę domkniętości świata, można wywnioskować:

(3) uczy-się-angielskiego (Jan)
(4) uczy-się-niemieckiego (Jan)

co natychmiast prowadzi do sprzeczności z (2).

LOGIKA DOMNIEMAŃ

Logika domniemań (ang. default logic) została wprowadzona w [28]. Służy ona do modelowania jednej z postaci wnioskowania naturalnego, zwanej **wnioskowaniem przez domniemanie** (ang. default reasoning). Wnioskowanie przez domniemanie jest to wyprowadzanie wiarygodnych, choć nie absolutnie pewnych hipotez z niepełnej informacji przy braku dowodu, że jest inaczej. Charakterystyczną cechą wnioskowania przez domniemanie jest to, że wyprowadzona konkluzja zawsze opisuje sytuację statystyczną lub prototypową.

Typowym przykładem wnioskowania przez domniemanie jest reguła o dzieciach i rodzicach. Jeśli Janek jest dzieckiem, to przy braku informacji, że jest inaczej, wnioskuje się, że ma on rodziców. Hipoteza ta oparta jest na obserwacji, że dzieci na ogół mają rodziców, większość dzieci ma rodziców lub że typowe dziecko ma rodziców.

W logice domniemań reguła o dzieciach i rodzicach jest reprezentowana przez następującą regułę wnioskowania, zwaną domniemaniem (ang. default):

$$\text{(1)}\quad \frac{\textbf{dziecko (x) : M ma-rodziców (x)}}{\textbf{ma-rodziców (x)}}$$

Regułę (1) interpretuje się następująco: dla każdego obiektu **x**, jeśli **x jest dzieckiem** i założenie, że **x ma rodziców**, jest niesprzeczne z posiadaną informacją, to należy wnioskować, że **x ma rodziców.**

Jeśli nasza wiedza o Janku sprowadza się do informacji, że Janek jest dzieckiem, to założenie, że Janek ma rodziców jest niesprzeczne z naszą wiedzą. Stosując zatem regułę (1) można wywnioskować, że Janek ma rodziców. Z drugiej strony, jeśli wiadomo, że Janek jest sierotą, to

[1]) Znak $\supset$ oznacza implikację.

założenie, że Janek ma rodziców jest sprzeczne z naszą wiedzą i reguła (1) nie może być zastosowana.

Niech **A** będzie formułą logiki pierwszego rzędu zawierającą zmienną x. Mówimy, że **x jest zmienną wolną w A**, jeśli x nie znajduje się w obrębie działania kwantyfikatora $\forall x$ lub $\exists x$. Na przykład formuła:

$$P(x) \vee \exists y Q(y, z)$$

zawiera dwie zmienne wolne: x oraz z. Dla podkreślenia, że $x_1, ..., x_n$ są zmiennymi wolnymi występującymi w formule **A**, formułę tę będziemy czasami oznaczać $A(x_1, ..., x_n)$.

W logice domniemań wiedza o świecie jest reprezentowana przez **teorię domniemań** (ang. default theory), tzn. parę składającą się ze zbioru aksjomatów zapisanych w klasycznej logice pierwszego rzędu i zbioru domniemań. Aksjomaty reprezentują niepodważalne fakty zachodzące w opisywanym świecie. Domniemania rozszerzają tę wiedzę o wiarygodne, choć niezupełnie pewne hipotezy.

Następujący przykład ilustruje prostą teorię domniemań składającą się z dwóch aksjomatów i dwóch domniemań:

(2) **ptak (Sylwia)**
(3) **dziecko (Janek)**
(4) $\dfrac{\textbf{ptak (x) : M lata (x)}}{\textbf{lata (x)}}$
(5) $\dfrac{\textbf{dziecko (x) : M ma-rodziców (x)}}{\textbf{ma-rodziców (x)}}$

Należy podkreślić, że w logice domniemań, jak i w innych formalizmach niemonotonicznych, niepodważalne fakty i obalalne hipotezy są traktowane jednakowo. Noszą też wspólną nazwę hipotez.

Formalnie domniemanie jest dowolnym wyrażeniem postaci:

$$\frac{A(x_1, ..., x_n) : M B(x_1, ..., x_n)}{C(x_1, ..., x_n)}$$

gdzie **A**, **B** i **C** są formułami klasycznej logiki pierwszego rzędu zawierającymi zmienne wolne $x_1, ..., x_n$. **A** nazywamy **warunkiem wstępnym** (ang. prerequisite), **B** — **uzasadnieniem** (ang. justification), a **C** — **wnioskiem** (ang. consequent) domniemania. Powyższe domniemanie ma następującą interpretację:

dla dowolnych obiektów $x_1, ..., x_n$, jeśli $A(x_1, ..., x_n)$ jest hipotezą i jeśli $B(x_1, ..., x_n)$ jest niesprzeczne ze zbiorem hipotez, to $C(x_1, ..., x_n)$ należy traktować jako hipotezę.

Domniemanie zawierające zmienne wolne nazywa się otwartym, a bez nich — zamkniętym. Domniemanie o dzieciach i rodzicach jest przykładem domniemania otwartego. Następujące wyrażenie:

$$\frac{\textbf{ptak (Sylwia) : M lata (Sylwia)}}{\textbf{lata (Sylwia)}}$$

jest przykładem domniemania zamkniętego.

O ile domniemania zamknięte są konkretnymi regułami wnioskowania, o tyle domniemania otwarte reprezentują pewne schematy wnioskowania, które mogą być użyte dla różnych konkretnych obiektów.

Należy odróżniać domniemanie otwarte od jego **konkretu** (ang. instance). Konkretem domniemania otwartego **D** jest dowolne domniemanie zamknięte, powstające z **D** wskutek zastąpienia wszystkich zmiennych wolnych nazwami konkretnych obiektów.
Na przykład domniemanie:

$$\frac{\textbf{ptak (Sylwia) : M lata (Sylwia)}}{\textbf{lata (Sylwia)}}$$

jest konkretem domniemania:

$$\frac{\textbf{ptak (x) : M lata (x)}}{\textbf{lata (x)}}$$

Teorię domniemań zawierającą przynajmniej jedno domniemanie otwarte nazywa się teorią **otwartą**. W przeciwnym wypadku mówi się, że teoria jest **zamknięta**.

Zbiór wszystkich hipotez wyprowadzalnych z danej teorii domniemań nazywa się **rozszerzeniem** (ang. extension) tej teorii. Pojęcie rozszerzenia definiuje się bezpośrednio tylko dla zamkniętych teorii domniemań. Jeśli **A** jest otwartą teorią domniemań. to jej rozszerzenie definiuje się jako rozszerzenie **zamkniętej teorii A'**, otrzymanej z **A** w wyniku pewnej transformacji. Transformacja ta sprowadza się do zastąpienia każdego domniemania otwartego występującego w **A** zbiorem jego wszystkich konkretów.

Powyższą transformację zilustrowano przykładem. Rozważmy następującą teorię otwartą:

(6) **ptak (Sylwia)**
(7) **słoń (Klaudiusz)**
(8) $\dfrac{\textbf{słoń (x) : M szary (x)}}{\textbf{szary (x)}}$

Ponieważ teoria ta zawiera dwa obiekty, można utworzyć dwa konkrety domniemania (8). A zatem teoria zamknięta odpowiadająca powyższej teorii wygląda następująco:

(9) **ptak (Sylwia)**
(10) **słoń (Klaudiusz)**
(11) $\dfrac{\textbf{słoń (Sylwia) : M szary (Sylwia)}}{\textbf{szary (Sylwia)}}$
(12) $\dfrac{\textbf{słoń (Klaudiusz) : M : szary (Klaudiusz)}}{\textbf{szary (Klaudiusz)}}$

Należy zwrócić uwagę, że warunek wstępny domniemania (11) nie jest hipotezą. A zatem domniemanie to nie będzie mogło być uaktywnione. Tym niemniej jest ono intuicyjnie akceptowalne. Gdyby Sylwia była słoniem, to fakt, że Sylwia jest szara, byłby zupełnie wiarygodną hipotezą.

Można teraz przejść do podania definicji rozszerzenia dla teorii domniemań. Ze względu na powyższą transformację, można ograniczyć się do teorii zamkniętych. Mówiąc niedokładnie, rozszerzenie teorii domniemań powinno zawierać wszystkie hipotezy wyprowadzalne z niej bądź przy użyciu logiki klasycznej, bądź w wyniku stosowania domniemań. Sugeruje to następującą definicję.

Niech $\mathbf{A} = (\mathbf{W}, \mathbf{D})$ będzie zamkniętą teorią domniemań, składająca się ze zbioru aksjomatów **W** i zbioru domniemań **D**. Zbiór formuł **E** jest rozszerzeniem teorii **A** wtedy i tylko wtedy, gdy **E** jest minimalnym zbiorem spełniającym następujące warunki:

(R1) $\mathbf{W} \subseteq \mathbf{E}$
(R2) $\mathbf{E} = Th(\mathbf{E})$
(R3) Dla każdego domniemania $(\mathbf{A} : \mathbf{MB} / \mathbf{C}) \in \mathbf{D}$, jeśli $\mathbf{A} \in \mathbf{E}$ oraz $\sim\mathbf{B} \notin \mathbf{E}$, to $\mathbf{C} \in \mathbf{E}$.

Warunek **(R1)** stwierdza, że każdy aksjomat jest hipotezą. Warunek **(R2)** stwierdza, że zbiór hipotez jest zamknięty ze względu na strukturę dedukcyjną klasycznej logiki pierwszego rzędu. Warunek **(R3)** stwierdza, że hipotezą jest konsekwencja każdego, możliwego do zastosowania domniemania.

Powyższa definicja ma charakter czysto ilustracyjny. Jest to uproszczenie definicji oryginalnej [28], która posługuje się pewnym operatorem algebraicznym i wykorzystuje pojęcie punktu stałego tego operatora.

Należy podkreślić, że istnieją teorie domniemań posiadające więcej niż jedno rozszerzenie. Poniżej podano przykład takiej teorii:

(13) **republikanin (Smith)**
(14) **kwakier (Smith)**
(15) $\dfrac{\textbf{republikanin (Smith) : M} \sim \textbf{pacyfista (Smith)}}{\textbf{pacyfista (Smith)}}$
(16) $\dfrac{\textbf{kwakier (Smith) : M pacyfista (Smith)}}{\textbf{pacyfista (Smith)}}$

Posiadana wiedza pozwala zastosować zarówno domniemanie (15), jak i (16). Z drugiej strony, użycie któregokolwiek z nich blokuje zastosowanie pozostałego. Dlatego powyższa teoria posiada dwa rozszerzenia. Jedno z nich zawiera hipotezę, że Smith jest pacyfistą, drugie — hipotezę przeciwną.

W wypadku, kiedy teoria domniemań posiada więcej niż jedno rozszerzenie, każde z nich jest traktowane jako alternatywny zbiór hipotez zachodzących w opisanym świecie.

Istotną wadą formalizmu Reitera jest to, że istnieją teorie domniemań nie posiadające rozszerzenia. Najprostszym, choć sztucznym przykładem, jest teoria składająca się wyłącznie z pojedynczego domniemania:

: M ~ pada-deszcz
pada-deszcz

Brak warunku wstępnego oznacza, że jest on nieistotny dla stosowalności domniemania. Przykład intuicyjnie naturalnej teorii domniemań, która nie posiada rozszerzenia, można znaleźć w [14]. W pracy tej podana jest również alternatywna formalizacja logiki domniemań. Istotną zaletą proponowanego podejścia jest fakt, że każda teoria domniemań posiada przynajmniej jedno rozszerzenie.

Pewne problemy związane z reprezentacją wiedzy przy użyciu logiki domniemań są dyskutowane w [7] i [29]. Semantyka logiki domniemań jest omawiana w [16]. Zastosowaniu logiki domniemań do przetwarzania języka naturalnego są poświęcone prace [6, 10, 23].

INNE FORMALIZMY NIEMONOTONICZNE

Hipoteza domkniętości świata i logiki domniemań nie wyczerpują wszystkich możliwych podejść do problemu formalizacji wnioskowania niemonotonicznego.

Jednym z ciekawszych formalizmów niemonotonicznych jest **logika niemonotoniczna** (ang. non-monotonic logic). Pierwszy system logiki niemonotonicznej podano w [22]. W logice niemonotonicznej domniemania są reprezentowane bezpośrednio w języku. Aby to było możliwe, język logiki niemonotonicznej zawiera specjalny operator modalny **M** interpretowany **niesprzeczne jest, że**. W logice niemonotonicznej regułę o dzieciach i rodzicach można zapisać w postaci następującego aksjomatu:
$\forall$ x **(dziecko (x)** $\wedge$ **M ma-rodziców (x)** $\supset$ **ma-rodziców (x))**
Pewne ogólne zasady związane z konstruowaniem logiki niemonotonicznej podano w [13]. Konkretne systemy logiki niemonotonicznej znaleźć można w pracach [1, 9, 15, 18, 21, 24].

Bardzo silnym mechanizmem niemonotonicznym jest **otaczanie** (ang. circumscription). Mechanizm ten wprowadzony został w pracy [19] i rozszerzony w [20]. Otaczanie zakłada reprezentację świata w języku logiki klasycznej. Niemonotoniczną strukturę dedukcyjną uzyskuje się na drodze czysto syntaktycznej, wskutek dołączenia pewnej formuły klasycznej logiki drugiego rzędu.

Główną wadą otaczania jest to, że jego niekontrolowane użycie może prowadzić do nieakceptowalnych hipotez. Co więcej, dość mało na razie wiadomo, jak używać tego formalizmu. Pewne ogólne właściwości otaczania przedstawiono w pracach [5, 8, 11, 12].

Innym podejściem do formalizacji wnioskowania niemonotonicznego jest rozszerzenie logiki standardowej o pewne mechanizmy metalogiczne. Podejście to zostało zaproponowane w [30] (zob. także [2] i [3]).

Formalizacja wnioskowania przez domniemanie przy użyciu trójwartościowej logiki monotonicznej została zaproponowana w [25]. Pełna bibliografia dotycząca wnioskowania niemonotonicznego do 1984 r. znajduje się w pracy [26]. Porównanie różnych podejść do wnioskowania niemonotonicznego, z punktu widzenia ich adekwatności do formalizacji wnioskowania naturalnego, można znaleźć w pracy [17].

LITERATURA

[1] Akama S.: Formalization of Non-Monotonic Logic from the Standpoint of Constructive Logic. Fujitsu Ltd., 1985
[2] Attardi G., Simi M.: Metalanguage and Reasoning Across Viewpoints. Proc. 6th Conf. ECAI, Pisa, 1984, pp. 315—324
[3] Bowen K. A., Kowalski R.: Amalgamating Language and Metalanguage in Logic Programming. Clark K. L., Tarnlund, S. A. (eds.), Logic Programming, APIC Studies in Data Processing No. 5, Academic Press, 1982
[4] Clark K. L.: Negation as Failure. Pp. 293—322, Gallaire H., Minker J. (eds.), Logic and Databases, Plenum Press, New York, 1978
[5] Doyle J.: Circumscription and Implicit Definability. AAAI Workshop on Non-Monotonic Reasoning, New York, 1984, pp. 57—69
[6] Dunin-Kęplicz B.: Default Reasoning in Anaphora Resolution Proc. 6th Conf. ECAI, Pisa, 1984, pp. 157—166
[7] Etherington D., Reiter R.: On Inheritance Hierarchies with Exceptions. Proc. AAAI-83, Conf. 1983, pp. 104—108
[8] Etherington D., Mercer R., Reiter R.: On the Adequacy of Predicate Circumscription for Closed-World Reasoning. AAAI Workshop on Non-Monotonic Reasoning, New York, 1984, pp. 70—81
[9] Gabbay D.: Intuitionic Basis for Non-Monotonic Logic, pp. 260—273. Proc. 6th Conf. CAD, Lecture Notes in Computer Science, Vol. 139, Springer, Berlin, 1982
[10] Joshi A., Webber B., Weischedel R.: Default Reasoning in Interaction. AAAI Workshop on Non-Monotonic Reasoning, New York, 1984, pp. 144—150
[11] Lifschitz V.: Some Results on Circumscription. AAAI Workshop on Non-Monotonic Reasoning, New York, 1984, pp. 151—164
[12] Lifschitz V.: Computing Circumscription. Proc. 9th Conf. IJCAI, Los Angeles, 1985, pp. 121—127
[13] Łukaszewicz W.: General Approach to Non-Monotonic Logics. Proc. 8th Conf. IJCAI, Karlsruhe, 1983, pp. 352—354
[14] Łukaszewicz W.: Considerations on Default Logic. AAAI Workshop on Non-Monotonic Reasoning, New York, 1984, pp. 165—193
[15] Łukaszewicz W.: Non-Monotonic Logic for Default Theories. Proc. 6th Conf. ECAI, Pisa, 1984, pp. 305—314
[16] Łukaszewicz W.: Two Results on Default Logic. Proc. 9th conf. IJCAI, Los Angeles, 1985, pp. 459—461
[17] Łukaszewicz W.: Formalization on Knowledge and Ignorance. The Journal for the Integrated Study of Artificial Intelligence, Cognitive Science and Applied Epistemology (w druku)
[18] Marek W.: A Natural Semantics for Modal Logic over Databases and Model-Theoretic Forcing I. AAAI Workshop on Non-Monotonic Reasoning, New York, 1984, pp. 194—240
[19] McCarthy J.: Circumscription — A Form of Non-Monotonic reasoning. Artificial Intelligence, No. 13, pp. 27—39, 1980
[20] McCarthy J.: Applications of Circumscription to Formalizing Common Sense Knowledge. AAAI Workshop on Non-Monotonic Reasoning, New York, 1984, pp. 295—324
[21] McDermott D.: Non-Monotonic Logic II/Non-Monotonic Modal Theories. Journal of the ACM, No. 29/1, pp. 33—57, 1982
[22] McDermott D., Doyle J.: Non-Monotonic Logic I. Artificial Intelligence, No. 13, pp. 41—72, 1980
[23] Mercer R., Reiter R.: The Representation of Presuppositions using defaults. Report 82-1, University of British Columbia, Vancouver, 1982
[24] Moore R.: Semantical Considerations on Non-Monotonic Logic. Proc. 8th Conf. IJCAI, Karlsruhe, 1983, pp. 272—279 (także: Artificial Intelligence, No. 25/1, pp. 75—94, 1984)
[25] Nutter J.: Default Reasoning Using Monotonic Logic: A Modest Proposal. Proc. AAAI, pp. 297—300, 1983
[26] Perlis D.: Bibliography of Literature on Non-Monotonic Reasoning. AAAI Workshop on Non-Monotonic Reasoning, New York, 1984, pp. 396—401
[27] Reiter R.: On Closed World Databases. Pp. 55—76; Gallaire H., Minker J. (eds.), Logic and Databases, Plenum Press, New York, 1978
[28] Reiter R.: A Logic for Default Reasoning. Artificial Intelligence, No. 13, pp. 81—132, 1980
[29] Reiter R., Criscuolo G.: On Interacting Defaults. Proc. 7th Conf. IJCAI, Vancouver, 1981, pp. 270—276
[30] Weyhrauch R.: Prolegomena to a Theory of Mechanized Formal Reasoning. Artificial Intelligence, No. 13, pp. 133—170, 1980.

Ceny ogłoszeń

Od 1 stycznia br. obowiązują następujące ceny ogłoszeń publikowanych na naszych łamach:
ogłoszenia duże (zależnie od objętości):
cała strona — 35 tys. zł, 3/4 — 30 tys., 1/2 — 25 tys., 1/4 — 20 tys., 1/8 — 15 tys.
ogłoszenia drobne (zależnie od liczby słów)
jedno słowo — 30 zł

Dodatki do ceny podstawowej:
— za dodatkowy kolor (na okładce) +30%
— za zamieszczenie ogłoszenia na czwartej stronie okładki +100%
— za zamieszczenie ogłoszenia na trzeciej stronie okładki +50%

Zniżki:
— za ogłoszenie 3—5-krotne —5%
— za ogłoszenia 6—10-krotne —10%
— za ogłoszenia 11-krotne i powyżej —20%
— za artykuły reklamowe i wkładki wykonane przez zleceniodawcę —40%
— za bloki i biuletyny wykonane przez zleceniodawcę — maks. —60%

W przypadku dostarczenia przez zleceniodawcę materiału ilustracyjnego nie odpowiadającego warunkom technicznym druku lub tekstu wymagającego redakcyjnego opracowania, do powyższych cen doliczane będą koszty odpowiednich usług fotograficznych, graficznych lub przygotowania tekstów.

PIOTR COFTA
Gdańsk

Czy następuje zmierzch języków asemblerowych?

Rozwój oprogramowania już od wielu lat dokonuje się głównie przez wykorzystanie języków wysokiego poziomu. Jest to zgodne zarówno z naturalnymi predyspozycjami człowieka, jak i z dążeniem do zwiększenia jakości i wydajności pracy programistów. Tym samym istotne staje się pytanie: czy nadszedł już zmierzch języków asemblerowych nie spełniających powyższych kryteriów, czy jest sensowne angażowanie jakiegokolwiek wysiłku w ich rozwój?

W artykule skrótowo przedstawiono historyczny rozwój języków asemblerowych, a także stan obecny oraz perspektywy ich rozwoju. Na tej podstawie spróbowano sformułować odpowiedź na pytanie zawarte w tytule.

Języki asemblerowe są niewątpliwie pierwszymi językami programowania stosowanymi w maszynach cyfrowych. Są one naturalnym rozszerzeniem możliwości programowania w języku maszynowym i czynią programowanie bardziej zrozumiałe dla człowieka. Pomimo ciągłego udoskonalania ich możliwości, względnie szybko zostały zdystansowane przez języki programowania wysokiego poziomu, choć dodano im takie właściwości, jak: definiowanie symboli, możliwość używania wyrażeń arytmetycznych czasu translacji, segmentacja programu, kompilacja dwuetapowa czy wreszcie strukturalizacja sterowania i danych.

HISTORIA I STAN OBECNY

Mimo tego atrakcyjność języków asemblerowych dla masowego użytkownika zmalała. Ich podstawowa cecha — tłumaczenie instrukcji na obraz pamięci przy zachowaniu pełnej kontroli programisty nad tym procesem — okazała się w tym wypadku nieistotna, a nawet przeszkadzała w programowaniu, zmuszając do pamiętania o szczegółach nieważnych z punktu widzenia użytkownika.

Obecnie, mając do wyboru wiele innych języków, języka asemblera używa tylko ten, dla kogo pewne zalety tego języka istotnie przewyższają trudności w posługiwaniu się nim. Oczywiście, uwaga ta jest słuszna tylko w sytuacji, w której użytkownik może swobodnie wybierać między wieloma językami i systemami programowania. Chociaż języki asemblerowe są stosunkowo trudne w użyciu, procent prac programistycznych, w których ich użycie uznano za niezbędne, jest nadal bardzo wysoki. Analiza [11] wykazała, iż w 1985 r. języki asemblerowe były używane na terenie USA w ponad 70% prac, a na 1986 r. planowano ich użycie w 50%. Należy przy tym zauważyć, iż w owych „straconych" 20% prac zdecydowano się na używanie języka C, będącego również rodzajem języka poziomu asemblera. W porównaniu z innymi językami rola języków asemblerowych nadal pozostaje znacząca (np. powszechność użycia Cobolu zmniejszyła się z 15 do 10% prac, Fortranu — z 40 do 25%, a Ady zwiększyła się z 5 do 10%).

Mgr inż. PIOTR COFTA w 1980 r. ukończył studia na Wydziale Elektroniki Politechniki Gdańskiej, o kierunku Informatyka. Pracuje jako starszy asystent w Instytucie Informatyki Politechniki Gdańskiej. Jego główne zainteresowania zawodowe skupiają się wokół problematyki produkcji translatorów, ze szczególnym uwzględnieniem języków asemblerowych.

Do tradycyjnych zalet programów tłumaczących z języka asemblerowego, tj. asemblerów [1]), zaliczano zazwyczaj: optymalność kodu, minimalizację zajmowanego obszaru pamięci, możliwość wykorzystania wszystkich właściwości komputera oraz pełną kontrolę nad procesem tworzenia przekładu. Należy w związku z tym rozpatrzyć, czy nadal te zalety dotyczą asemblerów, a jeżeli tak, to jakim grupom użytkowników są one obecnie potrzebne.

Optymalność kodu, w sytuacji ciągłego zmniejszania kosztów obliczeń, jest raczej wątpliwą zaletą. Dodatkowo rozwój kompilatorów optymalizujących powoduje, że niewielu użytkowników próbuje przyspieszyć działanie swojego programu przez umieszczenie w nim fragmentów napisanych w języku asemblera. Czyni się tak jedynie w wypadku obliczeń o bardzo dużej złożoności lub w wypadku systemów profesjonalnych, a wstawki takie wymagają wysokich kwalifikacji programisty. Ciekawostką jest fakt, iż pisząc programy dla współczesnych superkomputerów, konieczne jest pamiętanie o ich architekturze wewnętrznej, co czyni to programowanie bardzo zbliżonym pojęciowo do pracy na poziomie asemblera. Można nawet powiedzieć, że komputery te są programowane w pewnego rodzaju języku asemblerowym bardzo wysokiego poziomu, właśnie w celu optymalizacji kodu. Wymaga to jednak także programisty o wysokich kwalifikacjach.

Minimalizacja zajmowanego obszaru pamięci nie jest obecnie aż tak istotna dla osób korzystających ze sprzętu profesjonalnego, lecz jest nadal cenną właściwością dla użytkowników mikrokomputerów. Ponieważ każdy program rozrasta się z czasem do całej pamięci maszyny, wielu użytkowników ograniczonych możliwościami sprzętu lub używanych translatorów dąży do przekroczenia tej bariery głównie przez quasi-asemblerowe rozwiązania. Zapotrzebowanie na narzędzia służące do operowania pamięcią bez kontroli języków wysokiego poziomu skłania nawet twórców translatorów do tworzenia takich rozszerzeń, niezgodnych z duchem języka, lecz będących ukłonem w stronę użytkowników.

Możliwość wykorzystania wszystkich właściwości komputera, niedostępnych normalnie rozkazów, trybów pracy itp., jest atrakcyjna dla dwóch bardzo odległych od siebie grup użytkowników. Są to z jednej strony amatorzy, hobbiści, a z drugiej — profesjonaliści, twórcy nietypowych fragmentów oprogramowania. Tych pierwszych pociąga możliwość wykorzystania wszystkich możliwości najczęściej własnego sprzętu, realizacji nietypowych trików itp. Drugim możliwości te są potrzebne do projektowania fragmentów systemów operacyjnych, testów, monitorów itp.

[1]) Choć w literaturze anglojęzycznej i polskiej termin assembler występuje w różnych znaczeniach, za poprawne — szczególnie w tym samym tekście — można uznać tylko następujące: assembler (ang. assembler) — program dokonujący asemblacji; język asemblera, język asemblerowy (ang. assembly language, assembler language) — język, w którym wyrażone są asemblowane programy. (przyp. red.)

Możliwość programowania czystego komputera jest atrakcyjna dla studentów poznających maszyny cyfrowe na poziomie rejestrów, a także dla amatorów poznających własne mikrokomputery. W tym zastosowaniu asembler jest rodzajem pomocy dydaktycznej, w której pełna kontrola procesu wypełniania pamięci programem, jest rzeczą bardzo ważną.

Jak widać, zalety asemblerów są nadal cenione, lecz przez coraz węższą grupę użytkowników. Obecnie są to informatycy-specjaliści oraz amatorzy, właściciele mikrokomputerów. Należy zauważyć, że o ile pierwsza grupa powstała wskutek zmniejszenia się istniejącej poprzednio wielkiej grupy informatyków używających asemblerów, to druga stanowi nowość w rozwoju informatyki, włączając w zakres użytkowników ludzi, którzy często nie mają żadnego zawodowego związku z informatyką.

Z przedstawioną wyżej sytuacją związane jest zanikanie „wielkich asemblerów", typu MACRO-11 (dla komputerów PDP-11), o wielu wyrafinowanych możliwościach, licznych dyrektywach, trudnych w nauce i posługiwaniu się. Przez znaczny okres spełniały one rolę wzorca wymagań stawianych asemblerom, a także dzisiaj w istotny sposób wpływają na kierunek ich ewolucji, będąc punktem odniesienia dla każdej konstrukcji współczesnego asemblera. Coraz częściej asemblery te są wykorzystywane tylko jako narzędzia do wytwarzania asemblerów skrośnych czy też wyspecjalizowanych pakietów makrodefinicji. Jako odpowiedzi na potrzeby współczesnych użytkowników asemblerów należy oczekiwać rozwoju odpowiadających im wyspecjalizowanych narzędzi.

STANDARYZACJA

Jedną z podstawowych trudności w rozwoju języków asemblerowych jest niejednolitość postaci programów źródłowych i wynikowych, stwarzająca użytkownikom znaczne kłopoty zarówno w trakcie nauki, jak i pracy. W praktyce zmiana procesora i wiążąca się z tym zmiana języka asemblera wymaga obecnie nie tylko nauczenia się nowego zestawu instrukcji maszynowych, lecz również — nowych mnemonik, nowego sposobu zapisu instrukcji itp. Skrajnym przykładem jest mikroprocesor Z80, dla którego istnieją aż trzy różne warianty zapisu mnemonik. W tej sytuacji większość producentów oprogramowania podjęła próby pewnej lokalnej normalizacji, głównie pod kątem uproszczenia produkcji translatorów. Poza firmami działania te podążały dwoma torami. Jeden z nich stanowiły prace związane z próbą normalizacji opisu funkcji procesora [10], które obecnie — ze względu na rozwój nietypowych struktur procesorów, trybów adresowania itp. — nie doczekały się jeszcze widocznych efektów.

Drugi kierunek prac polega na dążeniu do ujednolicenia sposobu zapisu postaci źródłowych programów pisanych w językach asemblerowych. W pracach tych abstrahuje się od elementów zależnych od procesora, skupiając uwagę na uczynieniu postaci zewnętrznej języka bardziej jednolitą i zrozumiałą. Obecnie amerykańska organizacja IEEE zajmuje się tym problemem pod kątem normalizacji postaci zewnętrznej, kodu wynikowego, a także sposobu zapisu do pamięci. Prace te zostały uwieńczone opracowaniem kilku norm, m. in. [5]. W normie tej zaznaczył się — obok elementów oryginalnych — wyraźny wpływ „wielkich asemblerów", a także norm lokalnych, akceptowanych przez duże firmy.

Istnienie norm pozwala na łatwiejsze poznanie języka asemblera, sposobu notacji oraz sposobu korzystania z jego rozszerzeń. Ponadto umożliwia uproszczenie procedury wytwarzania translatorów, tak jak uczyniło to już wiele firm dla swoich produktów. W efekcie może to spowodować znaczne obniżenie ceny programów-translatorów oraz znaczne zwiększenie ich dostępności.

SPECYFIKA TRANSLATORÓW

Właściwości wielu języków asemblerowych są ściśle związane z możliwościami programów tłumaczących, tj. asemblerów. Poniżej omówiono te powiązania dla takich rodzajów programów, jak: asemblery strukturalne, makroprocesory, współasemblery i asemblery wbudowane.

Asemblery strukturalne

Dla potrzeb twórców systemów operacyjnych, profesjonalnych pakietów oprogramowania, testów itp., powstała grupa języków asemblerowych, które — ze względu na swoje specyficzne cechy — zostały nazwane strukturalnymi (odpowiednie programy tłumaczące nazywa się asemblerami strukturalnymi). Łączą one w sobie zalety programowania strukturalnego z możliwością dotarcia do potrzebnych szczegółów konstrukcji komputera. W chwili obecnej niewątpliwie dominującą rolę odgrywa język C [8, 12], którego sukces — ściśle związany z sukcesem systemu operacyjnego Unix — jest przesądzony. Mniej znany, choć godny uwagi, jest strukturalny język asemblerowy PLASYD [7], przeznaczony dla komputerów ODRA 1300. Jest rzeczą interesującą, że są one reprezentantami dwóch skrajnie przeciwnych podejść do konstruowania języków tego rodzaju. Język C powstał przez dopasowanie struktur właściwych językom wysokiego poziomu do potrzeb programowania na poziomie bliskim rzeczywistemu komputerowi. Stąd też wynika jedna z jego podstawowych zalet — przenośność oprogramowania.

W języku PLASYD starano się uogólnić pewne konstrukcje językowe, powtarzające się w programach pisanych w języku asemblera, zachowując przy tym możliwość wykorzystania przez użytkownika wszelkich właściwości komputera. Otrzymany język nie jest zatem przenośny ani też nie umożliwia używania takich struktur danych, jakie są dopuszczalne w języku C, gwarantując w zamian rzecz nieosiągalną w C, tj. rzeczywistą pełną kontrolę nad komputerem i procesem tłumaczenia.

Makroprocesory

Makroprzetwarzanie, a raczej język czasu translacji (czyli symbole, wyrażenia czasu translacji, mechanizm makroprzetwarzania i kompilacji warunkowej) stanowi najbardziej charakterystyczny element koncepcji „wielkich asemblerów". W praktyce, wskutek stopniowego odchodzenia od tej idei, nie jest oferowany obecnie bardziej wyrafinowany mechanizm niż ten, który został zaproponowany w MACRO-11 czy też w M 80. Co prawda, makroasemblery dla procesorów 16-bitowych mają pewne dodatkowe możliwości, jednakże w zasadzie są to dość nieznaczne modyfikacje i rozszerzenia tej samej koncepcji.

W większości obecnie produkowanych makroasemblerów dąży się do uproszczenia mechanizmu makroprzetwarzania, nawet za cenę ograniczenia jego możliwości. W istocie bowiem okazało się, iż owe bardzo wyrafinowane opcje makroprocesora były rzadko wykorzystywane. Spadek zainteresowania złożonymi mechanizmami makroprzetwarzania można również wiązać z uproszczeniem sposobu komunikacji programu z systemem operacyjnym, co było najczęściej obsługiwane przez pakiety makroinstrukcji. Również w kierunku pewnego uproszczenia tych mechanizmów podążają obecne prace normalizacyjne IEEE. Przykładowo, w projekcie normy dla makroasemblerów zrezygnowano z identyfikowania parametrów makroprzetwarzania przez nazwę, zastępując to prostszą identyfikacją przez numer kolejny. Podobnie w znacznym stopniu uproszczono mechanizm kompilacji warunkowej i zminimalizowano wymagania nałożone na wyrażenia arytmetyczne czasu translacji. Nowością zaproponowaną w projekcie normy jest całkowite odejście od przetwarzania „linia na linię" oraz wprowadzenie pojęcia funkcji czasu translacji.

Dotychczasowe makroprocesory zazwyczaj przetwarzały tekst według zasady: pełna linia makrowywołania na pewną liczbę pełnych linii makrorozwinięcia. Pierwsza próba zerwania z tą konwencją nastąpiła w makroasemblerach dla procesorów 16-bitowych [6]. W projekcie normy wprowadzono możliwość wymiany makrowywołania na część linii, a także możliwość zastępowania części linii (np. fragmentu wyrażenia arytmetycznego) rozwinięciem pewnego makrowywołania. Takie podejście gwarantuje możliwość o wiele bardziej elastycznego manipulowania tekstem w trakcie makroprzetwarzania.

Funkcja czasu translacji jest logicznym rozwinięciem i dopełnieniem języka czasu translacji. Jest ona deklarowana na podobieństwo makrodefinicji, lecz wywołana i obliczana (a raczej interpretowana) w trakcie obliczania wartości wyrażenia arytmetycznego. Istotną różnicą mię-

dzy wywołaniem funkcji a makrowywołaniem jest to, iż funkcja nie generuje żadnego tekstu, a jedynie udostępnia swoją obliczoną wartość (która być może jest potem zamieniana na tekst).

W tym kierunku podążają prace związane z tradycyjnie rozumianym makroprzetwarzaniem, przy czym oczywiście wiele istotnych szczegółów nie jest jeszcze uzgodnionych. Ponadto dużą popularność zdobywa koncepcja użycia pewnego języka wysokiego poziomu jako języka czasu translacji. Idea ta została niegdyś zastosowana w procesorze PL/I, jednakże w stosunku do asemblerów jej rozwój nastąpił dopiero wraz z rozwojem omówionych dalej asemblerów wbudowanych.

Współasemblery

Intensywnie rozwija się obecnie grupa współasemblerów przeznaczonych do realizacji krótkich prac w trybie interakcyjnym, np. tak jak w Basicu. Charakterystyczne dla tej techniki jest współistnienie w pamięci operacyjnej różnych postaci opracowywanego programu, głównie postaci źródłowej i binarnej, przy czym współasembler gwarantuje spojność wszystkich reprezentacji programu. System taki można użytkować również wtedy, gdy nie dysponuje się sprzętem wyposażonym w masową pamięć zewnętrzną. Dzięki połączeniu w jedną całość translatora, edytora i programu uzdatniającego (stąd też inna nazwa współasemblerów — TED, pochodząca ze złożenia pierwszych liter wyrazów: translator, editor i debugger) stanowią one doskonałe narzędzie do wprowadzenia niewielkich poprawek, badania pojedynczych instrukcji itp. Dodatkowe możliwości szybkiego łączenia ich z językami wysokiego poziomu, również w trybie interakcyjnym, zapewniają łatwe „łatanie" programu czy też inne podobne triki. Istnienie i rozwój współasemblerów są związane z makrokomputerami domowymi i osobistymi oraz z grupą użytkowników nieprofesjonalnych. Ponadto oczywiście, znajdują one zastosowanie w dydaktyce. Zazwyczaj języki akceptowane przez takie współasemblery nie są zbyt rozbudowane [9] (w sensie dyrektyw sterujących translacją itp.), jednak w niektórych rozwiązaniach oferowana jest nawet możliwość makroprzetwarzania [3]. Oczywiście, możliwości te są bardzo ograniczone, szczególnie w stosunku do „wielkich asemblerów" i rzadko kiedy obejmują więcej niż proste przekształcanie tekstu z możliwością parametryzacji czy też najprostsze konstrukcje translacji warunkowej.

Asemblery wbudowane

Z analizy przedstawionej w [11] wynika, że w znacznej części prac programistycznych używa się języka asemblera jako jednego z kilku języków potrzebnych do realizacji danego projektu. Stąd też niezwykle istotna jest możliwość łatwego łączenia tego języka z językami wysokiego poziomu. Tradycyjnie, tekst zapisany w języku asemblerowym jest tłumaczony na postać pośrednią, jednolitą dla wszystkich języków programowania. Następnie zaś, w procesie łączenia z takich fragmentów tworzony jest jednolity program w postaci gotowej do wykonania. Proces ten, choć zapewnia znaczne możliwości, np. tworzenie bibliotek, jest obecnie powolny i nieefektywny. Wszystkie języki programowania traktuje się w nim na tych samych prawach, poświęcając każdemu osobny etap przetwarzania. Tylko niektóre z języków odgrywają dominującą rolę w danym projekcie, a np. język asemblerowy służy zazwyczaj do tworzenia krótkich fragmentów programów. Takie rozwiązanie zwane równoprawnym, pomimo swoich braków, jest oczywiście nadal stosowane w różnych wariantach, zależnie od możliwości sprzętowych. Jako alternatywne pojawia się rozwiązanie bazujące na asemblerach wbudowanych.

Koncepcja asemblera wbudowanego nie jest nowa, a konstrukcje takie stosowano np. dla komputerów ODRA 1204. Jednak dopiero obecnie natrafiły one na rzeczywiste zapotrzebowanie ze strony programistów. Podstawową cechą asemblera wbudowanego jest ścisłe tekstowe zespolenie języka asemblerowego z pewnym językiem wysokiego poziomu. Oznacza to, iż fragmenty pisane w języku asemblerowym mogą być wkomponowywane w tekst programu i poprawiane w taki sam sposób i w tym samym czasie co główny tekst. Wiąże się z tym najczęściej pewne dopasowanie postaci źródłowej programów w języku asemblerowym do formatu języka macierzystego, podyktowane głównie chęcią zachowania jednolitej postaci tekstu źródłowego.

Najistotniejszym elementem rozróżniającym asemblery wbudowane jest moment translacji tekstu w języku asemblerowym w stosunku do translacji (lub interpretacji) tekstu napisanego w języku wysokiego poziomu. Najprostsza konstrukcja zastosowana np. w [2] polega na tym, że pewne procedury programu mogą być w całości napisane w języku asemblerowym. Być może, taka forma zapisu nie jest najwygodniejsza i bardziej eleganckie byłoby rozwiązanie zastosowane dla komputerów ODRA 1204, lecz koncepcja pozostaje ta sama. Polega ona na tym, że procedury napisane w języku asemblerowym są tłumaczone przez ten sam kompilator i w tym samym czasie, co procedury języka macierzystego. Prostota tego rozwiązania jest najczęściej okupiona brakiem możliwości tworzenia makrodefinicji, posługiwania się wyrażeniami czasu translacji itp., gdyż prowadziłoby to do nadmiernego rozbudowania samego kompilatora.

Bardzo ciekawym rozwiązaniem jest, zastosowane w [1], przeniesienie translacji fragmentów pisanych w języku asemblerowym na czas interpretacji (językiem macierzystym jest Basic) głównego programu. Realizowane jest to przez wplatanie zapisów asemblerowych między wiersze programu w Basicu, przy czym wykonywanie tekstu w Basicu jest równoważne jego interpretacji, a w chwili napotkania fragmentu napisanego w języku asemblera — następuje jego translacja. Wymaga to, co prawda, od programisty nieznacznie większej uwagi, zapewniając w zamian niezwykle elastyczny mechanizm wiążący oba języki. Przykładowo, o liczbie przebiegów translatora decyduje sam programista, zamykając stosowany fragment w pętlę **FOR**. Etykiety i symbole stosowane w zapisach asemblerowych są jednocześnie zmiennymi programu zapisanego w Basicu. Możliwe jest w tej sytuacji użycie wyrażeń arytmetycznych Basica jako wyrażeń czasu translacji, łącznie z procedurami i funkcjami. Z punktu widzenia języka asemblera procedury Basica spełniają rolę makrodefinicji, gdyż ich wywołanie może spowodować wygenerowanie odpowiedniego fragmentu programu w tym języku. Instrukcja **IF** i inne instrukcje sterujące określają możliwości kompilacji warunkowej. W ten sposób, niewielkim kosztem, asembler wbudowany zapewnia mechanizmy sterowania przebiegiem translacji bardziej wyrafinowane niż jakikolwiek makroprocesor, ponieważ w tym wypadku językiem czasu translacji jest wprost język Basic.

Innym przykładem, dobitnie świadczącym o tym, jak daleko może być posunięte wcielanie języka asemblerowego jest jego użycie w programach wyrażonych w języku Forth [4]. O ile w poprzednim rozwiązaniu tekst fragmentu napisanego w języku asemblera był tłumaczony w trakcie interpretacji programu podstawowego, to dla języka Forth tekst fragmentu jest zazwyczaj tłumaczony w czasie jego wykonania. Oznacza to, iż zapis w języku asemblera jest traktowany przez interpreter języka Forth jako ciąg komend, których wykonanie powoduje w efekcie zapisanie przekładu instrukcji asemblerowych w odpowiednim miejscu pamięci. Pozwala to użytkownikowi na jeszcze większą swobodę niż w poprzednim rozwiązaniu, bowiem nie tylko cały język Forth jest językiem czasu translacji, lecz ponadto nie istnieją prawie żadne przeszkody we wzajemnym przeplataniu się zapisów w języku asemblerowym z zapisami w języku macierzystym. Zwiększona swoboda powoduje znaczne zmniejszenie możliwości kontroli poprawności zapisu w języku asemblerowym, stąd też wymagana jest jeszcze większa uwaga programisty.

W tym wypadku upodobnienie się obu zapisów jest tak dalekie, że np. w języku asemblerowym wymaga się odwrotnej kolejności notacji, tj. najpierw podawania argumentów, a na końcu — mnemoniki instrukcji. Tak więc, konsekwencją stosowania asemblerów wbudowanych jest bardzo dokładne naśladownictwo (pod względem zapisu) składni języków macierzystych przez języki asemblerowe.

* * *

Języki asemblerowe utraciły swoją dominującą pozycję, rozumianą jako „najlepsze do wszystkiego". Stały się obecnie językami specjalistycznymi, stosowanymi raczej do prac uzupełniających. Jednocześnie zainteresowanie nimi nie maleje, istnieje nadal zapotrzebowanie na programistów znających języki tej klasy. Nie należy się spodziewać spadku zainteresowania nimi, gdyż nadal spełniają wiele istotnych funkcji, trudnych do zrealizowania innymi metodami. Ciekawe jest też pojawienie się nowej zainteresowanej nimi grupy — użytkowników nieprofesjonalnych.

Dalszy rozwój języków asemblerowych w dużej mierze uzależniony jest od ogólnego kierunku rozwoju informatyki. Można sądzić, że dalszy rozwój technik programowania jeszcze bardziej zredukuje sytuacje, w których asemblery są podstawowymi narzędziami pracy. Pozostaną natomiast asemblery wbudowane — jako szybkie narzędzia pomocnicze oraz współasemblery — traktowane jako pomoc dydaktyczna i narzędzie realizacji bardzo krótkich prac.

Istnienie strukturalnych języków asemblerowych jest związane z rozprzestrzenianiem się jedynego liczącego się przedstawiciela tej grupy, języka C. Choć jego popularność, związana z popularnością systemu operacyjnego Unix, wzrasta, nie należy spodziewać się powstawania nowych języków tego rodzaju.

Normalizacja wpłynie na potanienie i uproszczenie technik produkcji translatorów, lecz dotyczy to tylko tych języków asemblerowych, których translatory są samodzielnymi programami. Wpływ norm IEEE na asemblery wbudowane nastąpi wyłącznie w dziedzinie reprezentacji kodu źródłowego czy też pewnych elementów wyrażenia arytmetycznego.

Podsumowując, można zatem stwierdzić, że zmierzch języków asemblerowych nie nastąpił i nie nastąpi tak szybko, choć epoka ich wielkiego rozkwitu niewątpliwe przeminęła.

LITERATURA

[1] Acorn Computers: The Electron Users Guide, 1983
[2] Borland International: Turbo Pascal Reference Manual, 1984
[3] Bredon G. E.: Big Mac Macro Assembler — TED. Apple Pugetsound, 1981
[4] De Grandis-Harrison R.: Forth on the BBC Microcomputer. Acornsoft, 1983
[5] Institute of Electrical and Electronics Engineers: Standard for Microprocessor Assembly Language. IEEE Std. 694—1985
[6] Intel Corp.: MCS86 Macro Assembly Language Regerence Manual, 1979
[7] International Computers Ltd: Plasyd Users Guide. Document PLN-14, 1970
[8] Kerninghan B. W., Ritchie D. M.: The C Programming Language. Prentice-Hall, 1978
[9] Michalak B.: LIC — wspomaganie projektowania programów. Informatyka, T. 19, nr 7, str. 17—18, 1984
[10] Nicoud J. D.: CALM-2 — an Updated Common Assembly Language for 16- and 32-bit Microprocessors. Proc. EUROMICRO'80, North-Holland, 1980
[11] Schnidler M.: Through automation, software shapes itself to the task at hand. Electronic Design, Vol. 33, No. 17, pp. 87—98, 25 July 1985
[12] Winiewski J.: Język programowania C. Informatyka, nr 4—6, 1985.

ANDRZEJ GRZYWAK
JERZY KOŁODZIEJ
Instytut Systemów Sterowania
Katowice

Sieci komputerowe oparte na mikrokomputerach MERA 60

Sieci komputerowe powstały w wyniku prac nad systemami teleinformatycznymi, służącymi do bezpośredniego powiązania systemu komputerowego z miejscami powstawania informacji. W konsekwencji wyróżnia się pięć podstawowych etapów rozwoju sieci komputerowych:

- tworzenie systemu wielodostępnego w oparciu o jednostkę centralną i określoną liczbę terminali, najczęściej monitorów ekranowych (rys. 1a),
- zastąpienie terminala biernego terminalem inteligentnym, którego procesor realizuje dwie podstawowe funkcje: obsługę protokołu transmisji oraz lokalne przetwarzanie danych (rys. 1b),
- utworzenie tzw. lokalnej sieci hierarchicznej cechującej się dużą szybkością transmisji danych (rys. 1c),
- utworzenie sieci lokalnej z rozproszonymi zasobami (rys. 1d),
- utworzenie tzw. sieci otwartych obejmujących swym zasięgiem duże obszary (kraj, kontynent) i umożliwiających łączenie różnych typów komputerów (rys. 1e).

Obszary zastosowań sieci komputerowych w systemach przemysłowych

Rozwiązania techniczne	Obszary zastosowań
System teleinformatyczny (rys. 1a)	Wprowadzanie danych rozproszonych, scentralizowane systemy dyspozytorskie
Systemy z terminalami inteligentnymi (rys. 1b)	Rozbudowane systemy dyspozytorskie obejmujące duże rejony (np. rejon energetyczny)
Hierarchiczne sieci lokalne (rys. 1c)	Przemysłowe systemy dyspozytorskie, rozproszone systemy automatyzacji prac inżynierskich, systemy pomiarowe
Sieci lokalne typu magistralowego (rys. 1d)	Systemy kierowania i sterowania procesami technologicznymi, rozproszone systemy dyspozytorskie
Sieci otwarte (rys. 1e)	Duże rozproszone bazy danych do zarządzania branżą przemysłową

W zastosowaniach przemysłowych największe znaczenie mają sieci hierarchiczne z terminalami inteligentnymi i sieci lokalne obu rodzajów. W tabeli przedstawiono zakresy zastosowań poszczególnych rozwiązań sieci komputerowych w systemach przemysłowych.

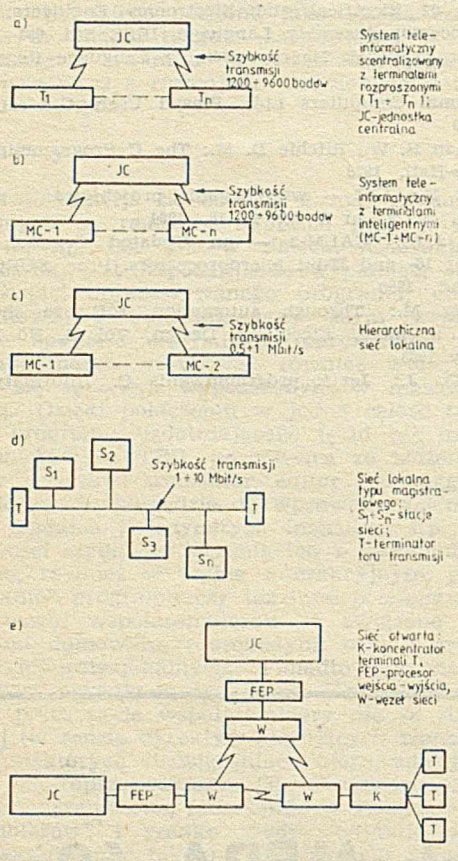
Przedmiotem niniejszego artykułu są rozwiązania sprzętowe i programowe sieci opartych na sprzęcie produkcji polskiej typu SM (MERA 60, SM 1633).

Sieć hierarchiczna z terminalami inteligentnymi

Rozwiązanie sieci hierarchicznych JS i SM wymaga zastosowania wielu urządzeń przeznaczonych do transmisji danych w trybie asynchronicznym lub synchronicznym. Komputer centralny, np. R-32, musi być wyposażony w procesor telekomunikacyjny z wyjściami przystosowanymi do transmisji szeregowej. Mikrokomputery dołączone jako terminale inteligentne muszą być wyposażone w moduły transmisji szeregowej asynchronicznej lub synchronicznej.

W momencie rozpoczęcia produkcji mikrokomputera SM 1633 (MERA 60) zdawano sobie sprawę, iż ważnym obszarem jego zastosowań będzie przetwarzanie rozproszone. Z tego też względu już pierwsze egzemplarze mikrokomputera wyposażono w moduły MDK-60, przeznaczone m. in. do dołączenia urządzeń przez linie telefoniczne. Moduł ten ma możliwość pracy z jednym modemem w trybie asynchronicznym, według niepełnego protokołu V-24. Kolejne moduły komunikacyjne spełniają coraz więcej funkcji. Moduł MTT-60 wykorzystuje prawie wszystkie możliwości standardu V-24 dla transmisji asynchronicznej, tj. pełny

dupleks, 16 różnych szybkości ustawionych programowo (do 19 200 bitów na sekundę), zmienną liczbę bitów danych (5..8), pracę w kanałach docelowych i powrotnych, programowy dostęp do sygnałów sterujących modemem. Obecnie w zastosowaniach wymagających transmisji asynchronicznej najczęściej stosowany jest moduł MMT-60, pozwalający na równoczesną obsługę czterech kanałów.



Rys. 1. Etapy rozwoju systemów teleinformatycznych i sieci komputerowych

Do współpracy ze standardowymi urządzeniami komputerów JS opracowano moduł transmisji synchronicznej MTS-60. Jest on wykorzystywany do komunikacji z użyciem protokołu BSC-3 i wyposażony w sprzętową realizację elementów tego protokołu (automatyczny nasłuch, wysyłanie sekwencji adresującej). Umożliwa także pracę na liniach wielopunktowych z szybkością transmisji do 9600 bit/s, z użyciem kodów EBCDIC lub ASCII.

Mikrokomputer SM 1633 (MERA 60) w większości zastosowań pracuje pod nadzorem systemów operacyjnych RT-60 V03 lub RT-60 V04. Dla obu tych praktycznie zgodnych wersji opracowano generator handlerów transmisji według protokołu BSC. Zależnie od zastosowania istnieje możliwość przygotowania handlera pracującego bądź według kodu EBCDIC głównie do współpracy z komputerami Jednolitego Systemu bądź też ASCII. W zależności od zastosowań handler może również w różny sposób obliczać sumę kontrolną. W oparciu o ten handler opracowano emulatory urządzeń jednolitego systemu.

Emulator urządzenia IBM 3780 emuluje czytnik kart, perforator kart i drukarkę. W rzeczywistości, do zapisywania i odczytywania danych wykorzystane są pamięci dyskowe. Dane przeznaczone do wysłania pochodzą z systemu zbierania danych. Dane są odbierane, a następnie drukowane przez specjalny program, który:

— przekodowuje dane z kodu EBCDIC na kod ASCII,
— drukuje dane w pożądanym formacie (w celu zmniejszenia liczby transmitowanych danych są one przesyłane w postaci skompresowanej wraz ze specyfikacją formatu wydruku).

Emulacja systemu zdalnych monitorów IBM 3270 jest realizowana za pomocą dwóch programów: wysokopriorytetowego — do obsługi rozkazów i poleceń przychodzących od komputera nadrzędnego i niskopriorytetowego — do obsługi funkcji lokalnych. Emulator obsługuje cztery monitory i drukarkę, dzięki czemu za pomocą monitorów mikrokomputera SM 1633 (MERA 60) można uzyskać bezpośredni dostęp do danych i programów komputera JS.

W aktualnych zastosowaniach emulator 3780 współpracuje z programami komputera JS pracującymi w systemie operacyjnym CRJE, natomiast emulator 3270 współpracuje z programami opartymi na metodzie dostępu typu BTAM.

Dla przygotowania danych przenaczonych do pracy wsadowej opracowano oprogramowanie automatycznej stacji zbierania danych. Jej rdzeniem jest program o nazwie SEE, który pozwala na zbieranie danych z czterech stanowisk. W programie tym zastosowano wiele algorytmów kontroli zapewniających praktyczną bezbłędność wprowadzanych danych. Kontrolowane są m. in. następujące parametry:

— przynależność znaku do określonego zbioru,
— zawieranie się wprowadzonej liczby w określonym przedziale,
— narastanie wartości pola w kolejnych rekordach.

Możliwe jest zdefiniowanie przez projektanta konkretnej procedury kontroli pól rekordu i zbioru rekordów. Należy podkreślić, że wprowadzane muszą być tylko istotne dane, gdyż inne mogą być powielane do wszystkich rekordów bądź też wprowadzane automatycznie podczas procedur kontrolnych na podstawie określonych zależności.

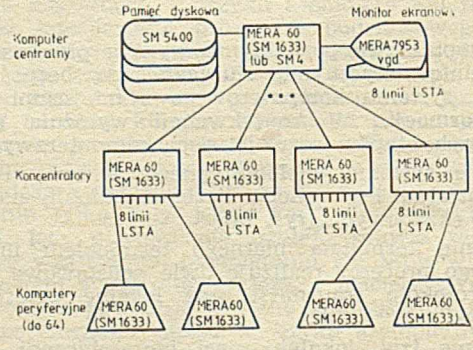
Oprogramowanie mikrokomputera SM 1633 umożliwa jego pracę w jednym z następujących trybów:

- jako stacji IBM 3780 do wprowadzania i wyprowadzania dużej liczby danych w reżimie pakietowym,
- jako stacji IBM 3270 do pracy interakcyjnej,
- jako stacji redagowania wydruku danych otrzymanych z głównego komputera w postaci skompresowanej (off-line).

W oparciu o przedstawione rozwiązania w Południowym Okręgu Energetycznym zrealizowany jest system dyspozytorski obejmujący swym zasięgiem kilka elektrowni.

Sieć lokalna hierarchiczna SN-60

Strukturę sieci SN-60 utworzonej z komputerów MERA 60 przedstawiono na rys. 2. Można w niej wydzielić trzy poziomy: poziom komputera centralnego, poziom koncentratorów i poziom komputerów peryferyjnych.



Rys. 2. Struktura sieci SN-60

Specyficzny charakter sieci SN-60 sprawia, że może ona być sklasyfikowana w następujący sposób:
— jako sieć lokalna, ze względu na fizyczny zasięg przesyłanych informacji, tzn. do 1 km,
— jako sieć jednorodna, ze względu na rodzaj sprzętu, jaki może pracować w sieci (komputery rodziny SM),
— jako sieć hierarchiczna, ze względu na konfigurację.

Systemy komputerowe pracujące w sieci SN-60 pozwalają użytkownikowi na:
— korzystanie ze wspólnych zasobów dyskowych,
— korzystanie ze wspólnych urządzeń zewnętrznych,
— komunikację między użytkownikami.

Komputerem centralnym może być minikomputer SM-4 lub mikrokomputer MERA 60 (SM 1633) wyposażony w pamięć dyskową oraz konsolę operatora. Komputer centralny zapewnia przechowywanie informacji na wszystkich nośnikach pamięci masowej. Podstawowym zadaniem komputera centralnego jest obsługa transmisji danych

i programów, jaka odbywa się pomiędzy komputerami peryferyjnymi a pamięciami masowymi.

Koncentratory zbudowane w oparciu o mikrokomputery MERA 60 służą do lepszego wykorzystania środków łączności w sieci SN-60. Uzyskuje się to dzięki dołączeniu kilku komputerów peryferyjnych — w pewnym ograniczonym obszarze przestrzennym — do jednego koncentratora, który z kolei łączy się za pośrednictwem jednej linii LSTA (Linia Szybkiej Transmisji Asynchronicznej wykonana w postaci pary przewodów koncentrycznych o impedancji falowej 50 lub 75 omów) z komputerem centralnym. Jeśli komputer peryferyjny występuje samodzielnie, to oczywiście może być dołączony bezpośrednio do komputera centralnego.

Komputery peryferyjne, którymi mogą być zarówno mikrokomputery MERA 60 jak i MERA 660, mają za pośrednictwem sieci SN-60 dostęp do wspólnych zasobów umieszczonych w komputerze centralnym. Ważnym elementem wyposażenia każdego komputera peryferyjnego jest program inicjujący (ang. bootstrap), który pozwala na automatyczne zainicjowania pracy komputera peryferyjnego po włączeniu zasilania. Komputer peryferyjny może być wyposażony we wszystkie urządzenia zewnętrzne wymagane przez użytkownika i może pełnić funkcje:

- zautomatyzowanego stanowiska pracy eksperymentatora, które przez magistralę CAMAC lub IEEE 488 pozwala na bezpośredni dostęp do obiektu,
- koncentratora terminali, który umożliwia opracowywanie programów lub sporządzanie dokumentów przez operatora poszczególnych końcówek,
- inteligentnego sterownika (jako mikrokomputer bez terminala), który wykonuje zawsze te same zadania sterowania obiektem za pomocą oprogramowania załadowanego z komputera centralnego.

W wymianie informacji w sieci SN-60 bierze udział pięć następujących warstw:

— warstwa sprzętowa,
— handlery linii,
— sieciowe programy usługowe,
— system operacyjny,
— oprogramowanie użytkowe.

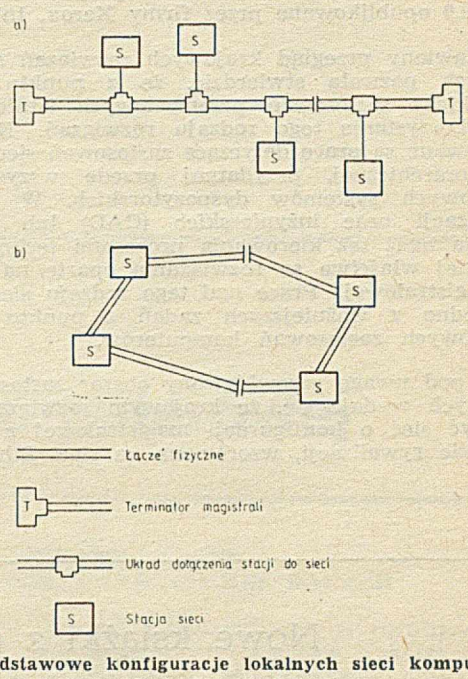
Warstwa sprzętowa jest realizowana całkowicie przez moduły szybkiej transmisji asynchronicznej (MAP-60). Moduły MAP-60 znajdują się na obu końcach linii LSTA i są typowymi modułami systemu MERA 60 (komputer SM-4 ma odpowiednik modułu MAP-60 dla magistrali UNIBUS). Pozostałe warstwy informacji są realizowane programowo.

Na zainteresowanie sieciami jednorodnymi typu SN-60 wpłynęły zarówno dotychczasowe doświadczenia w zakresie eksploatacji komputerów MERA 60, jak również sytuacja rynkowa, szczególnie jeśli chodzi o dostępność i ceny urządzeń peryferyjnych. Na rozwój sieci jednorodnych mają wpływ trudności sprzętowe i przestarzała technologia pracy, czego wyrazem są następujące elementy:

- stosunkowo drogie (w porównaniu z kosztami naszego procesora i półprzewodnikowej pamięci operacyjnej) i trudno dostępne na naszym rynku pamięci masowe o dużych pojemnościach,
- dość kosztowne i równie trudno dostępne urządzenia zewnętrzne o dobrych parametrach funkcjonalnych, jak np. urządzenia graficzne,
- niezbyt dogodny sposób wymiany danych i programów między użytkownikami pracującymi na autonomicznych systemach opartych na komputerach rodziny SM, polegający na fizycznym przenoszeniu nośników,
- niezbyt dogodny, przy tradycyjnym podejściu, dostęp do wspólnej bazy danych wymagający bezpośredniego kontaktu użytkownika z systemem, na którym jest założona i przechowywana wspólna baza danych.

Kierunki rozwoju sieci lokalnych

Przez lokalną sieć komputerową rozumie się zbiór co najmniej kilku połączonych systemów komputerowych, rozmieszczonych na niewielkim obszarze, wraz z zespołem środków sprzętowych i programowych, które umożliwiają tym komputerom wzajemną komunikację oraz korzystanie ze wspólnych zasobów. W praktyce, ze względów ekonomicznych, komunikacja między dowolną parę komputerów jest realizowana w oparciu o pojedyncze łącze łączące wszystkie komputery, udostępniane w całości (konfiguracja magistralowa — rys. 3a) lub odcinkami (konfiguracja pierścieniowa — rys. 3b) na zasadzie podziału czasu dla realizacji transmisji.



Rys. 3. Podstawowe konfiguracje lokalnych sieci komputerowych: a — sieć magistralowa, b — sieć pierścieniowa

Zasada uzyskiwania przydziału łącza jest określona stosowaną w sieci metodą (reguła) dostępu. W wypadku sieci konfiguracji magistralowej dominują rozwiązania umożliwiające dostęp do łącza na zasadzie rywalizacji (np. reguła CSMA/CD) lub na zasadzie przekazywania uprawnienia (metoda stosowana również w sieciach o konfiguracji pierścieniowej). Aktualnie brak jest podstaw do jednoznacznego stwierdzenia, która metoda dla sieci magistralowej jest lepsza. Rywalizacyjna reguła dostępu zapewnia minimalny narzut organizacyjny oraz natychmiastowy dostęp przy niewielkim obciążeniu łącza. Z drugiej strony, przy dużym obciążeniu (występowanie konfliktów przy dostępie do łącza) nie gwarantuje pomyślnej realizacji transmisji. Organizacja dostępu na podstawie przekazywania uprawnienia charakteryzuje się dużym, w przybliżeniu stałym niezależnie od obciążenia sieci, narzutem organizacyjnym. Ponadto, wprowadza konieczność oczekiwania na uzyskanie dostępu do łącza. Pozwala jednak określić maksymalny czas, po którym dostęp ten zostanie uzyskany. Wadą tej metody jest stosunkowo duży narzut organizacyjny związany z dołączaniem stacji do sieci i wyłączaniem jej z sieci.

Obecnie powstało wiele sieci eksperymentalnych charakteryzujących się bardzo różnorodnymi rozwiązaniami. Wśród rozwiązań oferowanych handlowo dominuje znane już z INFORMATYKI [1] rozwiązanie sieci o konfiguracji magistralowej z rywalizacyjną regułą dostępu, opracowane przez firmy Xerox, DEC i Intel pod nazwą Ethernet. Sieć ta jest przeznaczona głównie do automatyzacji prac biurowych, realizacji elektronicznej poczty oraz zdalnego dostępu do specjalizowanych urządzeń zewnętrznych i pamięci masowych. W rozwój handlowy tej sieci zaangażowanych jest wielu producentów sprzętu komputerowego (m. in. firma DEC w zastosowaniu do komputerów serii PDP-11 i VAX) i układów scalonych, oferujących układy umożliwiające dołączenie do sieci Ethernet różnorodnych komputerów, od osobistych do dużych maszyn cyfrowych, np. IBM 370. Oprócz sieci lokalnych do zastosowań biurowych oferowanych jest wiele rozwiązań sieci do sterowania procesami przemysłowymi. Najczęściej są to sieci o konfiguracji magistralowej z dostępem do łącza na zasadzie przekazywania uprawnienia.

Ze względu na różnorodność stale powstających nowych opracowań sieci lokalnych, stosunkowo wcześnie podjęto prace normalizacyjne, mające na celu zdefiniowanie standardowych rozwiązań dla różnych obszarów zastosowań.

Z kolei firmy, które pierwsze opracowały i wprowadziły swoje rozwiązania publikując dokładne specyfikacje, próbowały narzucić własne standardy. Najbardziej znana sieć Ethernet w wyniku prac normalizacyjnych została opisana w kilku dokumentach, jak: norma IEEE Std. 802, normy ECMA 80, 81, 82 (ISO/TC97/SC13), specyfikacje Ethernet V1.0 i V2.0 opublikowane przez firmy Xerox, Intel i DEC.

Przedstawiony przegląd krajowych rozwiązań sieci komputerowych pozwala stwierdzić, że z punktu widzenia pewnej klasy zastosowań przemysłowych istnieje możliwość wykorzystania tego rodzaju rozwiązań. Najbardziej zaawansowane są prace dotyczące zastosowań sieci o strukturze hierarchicznej, przydatnej przede wszystkim dla przemysłowych systemów dyspozytorskich. W systemach automatyzacji prac inżynierskich (CAD) lub systemach sterowania bądź też kierowania procesami technologicznymi bardziej właściwe są rozwiązania oparte na architekturze magistralowej. Prace nad tego rodzaju sieciami stanowią jedno z ważniejszych zadań z punktu widzenia przemysłowych zastosowań komputerów.

Biorąc pod uwagę przewidywany obszar zastosowań sieci lokalnych wydaje się, że krajowym rozwiązaniem powinna być sieć o konfiguracji magistralowej z dostępem na zasadzie rywalizacji, wzorowana na sieci Ethernet. Za przyjęciem takiego rozwiązania, oprócz specyficznych właściwości tej sieci, przemawia wiele innych czynników.

Sieć Ethernet jest stosowana i testowana od kilku lat; dostępne są materiały przedstawiające zarówno różne warianty konstrukcyjne jak i informacje eksploatacyjne, które powinny pomóc w uniknięciu pomyłek projektowych. W opracowaniu tej sieci brała udział firma DEC, której komputery serii PDP-11 wyposażane są przez wielu producentów w układy sprzęgające z Ethernetem — istnieje więc oprogramowanie realizujące funkcje wyższych warstw sieci. Bardzo ważny jest fakt, że dla sieci Ethernet są produkowane lub wchodzą do produkcji układy scalone LSI realizujące podstawowe funkcje sterownika sieci. Jest istotne również to, że materiały normalizacyjne dotyczące rozwiązań sieci Ethernet są dostępne z kilku źródeł. Dotychczas zrealizowano modelowe rozwiązania sieci magistralowej o nazwie Lokal SM. Sieć ta umożliwa połączenie komputerów SM 1633 (MERA 60) i SM-1300.

Nadesłano: 1986.08.22

LITERATURA

[1] Zakharov V.: Stan rozwoju lokalnej sieci komputerowej Ethernet. Informatyka, nr 9—12 1986.

Nowe książki z dziedziny sztucznej inteligencji

D. Michie: On Machine Intelligence (wydanie poprawione i rozszerzone). Ellis Horwood, 1986.

Książka zawiera przegląd i analizę aktualnie prowadzonych prac zmierzających do stworzenia maszyn inteligentnych. Przedstawiono w niej teoretyczną, matematyczną i lingwistyczną wiedzę na temat procesów poznawczych, które leżą u podstaw inteligencji maszynowej. Książka składa się z czterech niezależnych części: gry komputerowe (gra i plan, szachy komputerowe a humanizacja techniki), inteligentne roboty (zintegrowane systemy poznawcze, sztuczna inteligencja, maszyny a teoria inteligencji, inżynieria wiedzy), mechanika poznania (mechanizmy pamięciowe i uczenie maszynowe, maszynowe modele umiejętności percepcyjnych i intelektualnych, pomiary zawartości wiedzy w systemach ekspertowych, automatyczna synteza wiedzy) oraz sztuczna inteligencja a społeczeństwo (komputer — sługa czy pan, przyspieszenie zdobywania wiedzy, inteligencja maszynowa).

P. Bishop: Fifth Generation Computers — Concepts, Implementations and Uses. Ellis Horwood, 1986.

Celem książki jest przedstawienie aktualnego zakresu prac badawczo-rozwojowych zmierzających do stworzenia komputerów piątej generacji. Książka zawiera wyważone wprowadzenie do zagadnień sztucznej inteligencji oraz koncentruje się na omówieniu architektury, sprzętu, języków programowania, zagadnień inżynierii oprogramowania, zastosowań oraz tendencji rozwojowych komputerów piątej generacji. Podkreślono w niej znaczenie budowy inteligentnych systemów z bazami wiedzy oraz inteligentnych sprzęgów.

F. Holtz. LISP — The Language of Artificial Intelligence. Tab Books Inc., 1986.

Książka zawiera opis Lispu, przegląd jego podstawowych funkcji oraz wiele praktycznych wskazówek programistycznych. Zaletą książki jest ukierunkowanie na muLISP, dialekt Lispu powszechnie stosowany na mikrokomputerach. Podano liczne przykłady i programy, które mogą posłużyć za praktyczne wprowadzenie do programowania zadań sztucznej inteligencji.

H. De Saram: Programming in Micro-PROLOG. Ellis Horwood, 1985.

Autor demonstruje możliwości mikrokomputerowej wersji języka programowania logicznego Prolog, na przykładach z dziedziny sztucznej inteligencji. Pierwsza część zawiera wprowadzenie do micro-Prologu oraz opis typowych metod stosowanych w rozwiązywaniu zadań. Druga część omawia takie zastosowania, jak: gry i zagadki, rozszerzona grafika żółwia, bazy danych oraz przetwarzanie języka naturalnego. Po przeczytaniu książki użytkownik będzie w stanie rozwiązywać za pomocą mikrokomputera niebanalne zadania z zakresu sztucznej inteligencji.

D. A. Waterman: A Guide to Expert Systems, Addison-Wesley, 1986.

Celem książki jest ułatwienie wyboru odpowiedniego systemu ekspertowego spośród wielu programów tego rodzaju dostępnych na rynku. We wstępie wytłumaczono działanie systemów ekspertowych, celowość i metody ich budowy. sposoby uzyskiwania wiedzy od ekspertów oraz ograniczenia. Książka zawiera opisy ponad 200 systemów ekspertowych i narzędzi do ich konstruowania.

L. Brownston, R. Farrell, E. Kant: Programming Expert Systems in OPS5 — An Introduction to Rule-Based Programming. Addison-Wesley, 1985.

Książka prezentuje praktyczne metody programowania regułowego w zastosowaniu do systemów ekspertowych za pomocą OPS5. Pierwsza część książki zawiera opis języka OPS5 i uczy efektywnych technik programowania na przykładzie rozwiązania konkretnego problemu, począwszy od fazy definiowania zadania, a skończywszy na fazie testowania. Druga część książki zawiera przegląd różnych architektur systemów regułowych i porównuje OPS5 z innymi dostępnymi narzędziami.

C. Townsend, D. Feucht: Designing and Programming Personal Expert Systems. Tab Books Inc., 1986.

Autorzy podają pełny opis funkcjonującego systemu eksprtowego. System napisany w języku Forth z przeznaczeniem na mikrokomputery IBM PC i Apple II umożliwia budowę bazy wiedzy z 200—300 reguł. Na podstawie tego roboczego modelu autorzy omawiają metody konstrukcji systemów ekspertowych.

STANISŁAW GÓRAL
KRZYSZTOF JANCZEWSKI
Centrum Obliczeniowe
Politechnika Wrocławska
JÓZEF JANYSZEK
Ośrodek Obliczeniowy
Politechnika Wrocławska
ZBIGNIEW ZAJĄCZKOWSKI
Biblioteka Główna i OINT
Politechnika Wrocławska

Użytkowanie Międzyuczelnianej Sieci Komputerowej

W poprzednich artykułach [1, 3] omówiono główne składniki Międzyuczelnianej Sieci Komputerowej (MSK): podsieć komunikacyjną i komputer obliczeniowy. Podsieć komunikacyjną tworzą węzły połączone dzierżawionymi liniami telefonicznymi. Do węzłów dołączone są komputery obliczeniowe (koncentratory terminali), które wykorzystują podsieć do przesyłania informacji i danych między sobą i zarazem świadczą określone usługi użytkownikom. Na przykładzie komputera obliczeniowego ODRA 1305 pokazano, jakie mechanizmy programowe należało dodatkowo opracować, aby takie przesyłanie między dwoma komputerami obliczeniowymi było możliwe. Większą część tego oprogramowania umiejscowiono w komputerze czołowym (aby nie obciążać komputera obliczeniowego), a część — w komputerze obliczeniowym.

Oprogramowanie komputera czołowego steruje pracą terminali dołączonych do niego, umożliwia przesyłanie tekstów między dwoma dowolnymi terminalami w sieci (lokalnie też), zestawia połączenie między danym terminalem a dowolnie wybranym komputerem obliczeniowym (aktualnie z ODRĄ 1305, uruchamiany jest dostęp do R-32), umożliwia pracę na różnych typach terminali. Modułowa struktura oprogramowania komputera czołowego umożliwia wygenerowanie takiej wersji, aby komputer czołowy mógł pełnić, na przykład, tylko funkcje koncentratora terminali. Wprowadzenie komputera czołowego uniezależnia użytkownika od jego lokalnego komputera obliczeniowego.

Oprogramowanie komputera obliczeniowego rozszerzono o system transferu zbiorów i zadań (TZZ), a systemowy język opisu zadań — o nowe polecenia umożliwiające użytkownikowi korzystanie z usług systemu TZZ. Tak więc komputery obliczeniowe wyposażone w system TZZ mogą między sobą (na polecenie użytkownika wydane z terminala lub z zadania wsadowego) wymieniać dane, zadania i przesyłać dane z jednego komputera do urządzeń zewnętrznych innego komputera.

RETROSPEKTYWNY DOSTĘP DO BIBLIOGRAFICZNYCH BAZ DANYCH

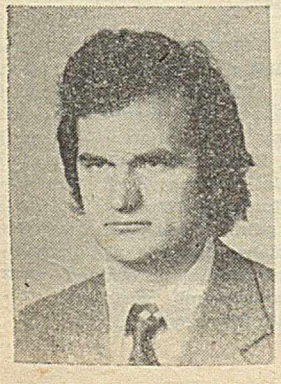
Powstanie sieci komputerowej sprzyja rozwojowi nowej usługi, jaką jest dystrybucja i szerokie wykorzystanie baz danych, a w szczególności retrospektywny dostęp do bibliograficznych baz danych. Bank danych staje się ekonomicznie efektywny dopiero wówczas, gdy korzysta z niego duża liczba abonentów. Takie możliwości stwarza sieć komputerowa.

Realizując retrospektywny dostęp do bibliograficznych baz danych w MSK opracowano system IDOL (Informacja Dokumentacyjna On-Line) [7].

Dialog z bazą danych w systemie IDOL prowadzi się za pomocą specjalnego prostego zestawu poleceń (komend) i pytań. Za pomocą pytań użytkownik określa tematykę i zbiór poszukiwanych opisów dokumentów. Polecenia umożliwiają wyprowadzenie wybranych opisów dokumentów na terminal lub drukarkę wierszową wskazanego komputera oraz służą do wykorzystania pozostałych możliwości systemu.

Pytania działają na słowniku bazy danych, który podzielono na kilka kategorii wyszukiwawczych. Każda kategoria jest identyfikowana przez jednoliterowy kod. Wyszczególniono osiem kategorii wyszukiwawczych:

A — autorzy,
C — kod jednoznacznie określający tytuł czasopisma,
D —deskryptory (jedno- i wielosłowowe terminy z tezaurusa serwisu informacyjnego),
L — język publikacji,
S — symbole klasyfikacji,
T — słowa (unitermy) wybrane z tytułu oraz słów kluczowych,
X — kod rodzaju publikacji,
Y — data publikacji.



Mgr inż. STANISŁAW GÓRAL w 1970 r. ukończył Wydział Elektroniki i rozpoczął pracę w Instytucie Organizacji i Zarządzania Politechniki Wrocławskiej. Od 1972 r. pracował w ZETO-Wrocław, gdzie m. in. był koordynatorem wdrożenia systemu wielodostępnego POLRAX-2. W 1977 r. otrzymał nagrodę ministra nauki, szkolnictwa wyższego i techniki. Od 1979 r. jest zatrudniony w Centrum Obliczeniowym Politechniki Wrocławskiej. Zajmuje się problemami eksploatacji i pomiarów sieci komputerowych, podejmuje także prace z zakresu systemometrii informatycznej.

Dr KRZYSZTOF JANCZEWSKI ukończył studia na Wydziale Mat.-Fiz-Chem. Uniwersytetu im. A. Mickiewicza w Poznaniu. W latach 1963—1972 był zatrudniony w Instytucie Matematyki, a od 1972 r. pracuje w Centrum Obliczeniowym Politechniki Wrocławskiej. Zajmuje się oprogramowaniem sieci komputerowych oraz wdrożeniem i badaniem pilotowej sieci MSK.

Pytania dzielą się na proste i złożone. Proste pytania mają postać:

xx K > termin
xx K:termin
xx K < termin

gdzie: xx jest identyfikatorem pytania złożonym z litery i cyfry, **K** — jednoliterowym kodem kategorii wyszukiwawczej słownika, **termin** — ciągiem znaków poszukiwanych w słowniku danej kategorii wyszukiwawczej.
Symbol oddzielający kod kategorii od terminu oznacza poszukiwanie terminu:
: — identycznego z podanym,
> — identycznego z podanym i większego (według kodu ICL),
< — identycznego z podanym i mniejszego (według kodu IGL).

W podanych terminach można stosować specjalne znaki #, ▫. Znak # może być umieszczony przed lub po terminie i oznacza, że przedrostek lub końcówka terminu mogą być dowolne. Znak ▫ oznacza natomiast maskowanie dowolnego znaku w terminie.

Pytania złożone służą do połączenia wcześniej zadanych pytań (prostych i złożonych) operatorami logicznymi. Można wykorzystywać pełną logikę boolowską (operatory sumy, iloczynu i negacji) oraz używać nawiasów.

Przykłady pytań prostych i złożonych:
A1 A : KOWALSKI — pytanie proste;
B1 T : NETWOR# — pytanie proste;
C1 = A1 * B1 — pytanie złożone (iloczyn);
B2 Y > 1983 — pytanie proste;
C2 = C1 * B2 — pytanie złożone.

W odpowiedzi na pytania system podaje liczbę opisów dokumentów znalezionych w bazie, zgodnych z warunkami zadanymi w pytaniu, a w wypadku pytania prostego dodatkowo informuje, ile terminów w słowniku systemu jest zgodnych z wzorcem podanym w pytaniu.

Polecenia systemu dzielą się na polecenia drukowane, wyprowadzania, ustawiania, zachowania oraz polecenia pełniące funkcje pomocy i kończenia sesji. Język poleceń jest prosty i przejrzysty. Operuje się dwuliterowymi mnemonicznymi symbolami poleceń. Ich pełny wykaz podano w tabeli.

Wykaz poleceń systemu IDOL

Symbol polecenia	Znaczenie
DD	Drukuj po sesji opisy dokumentów
DZ	Drukuj po sesji zakres przeszukiwanych części zestawu danych
KO	Kończ sesję
PO	Pomóż
UA	Ustaw adres
UI	Ustaw aktualne wartości parametrów instalacyjnych
UZ	Ustaw zakres przeszukiwanych części zestawu danych
WA	Wyprowadź na terminal adres
WD	Wyprowadź na terminal opisy dokumentów
WI	Wyprowadź na terminal wartości parametrów instalacyjnych
WP	Wyprowadź na terminal pytania
WS	Wyprowadź na terminal fragment słownika
WZ	Wyprowadź na terminal zakres dostępnych części zestawu danych
ZP	Zachowaj wskazane pytania

Użytkownik rozpoczyna pracę pisząc na terminalu IDOL (uruchamia makros o tej nazwie). System w odpowiedzi wyprowadza informacje, które jego administrator chce przekazać wszystkim użytkownikom, a następnie informuje, jaki numer został przyporządkowany użytkownikowi. W wypadku awarii systemu GEORGE-3 lub konieczności prowadzenia pracy przez użytkownika, znajomość tego numeru umożliwia wznowienie pracy i jej kontynuowanie (restart).

Za pomocą systemu IDOL mogą być udostępniane wszystkie bibliograficzne bazy danych sprowadzane do Polski z zagranicy i wykorzystywane dotychczas jedynie w trybie selektywnej dystrybucji informacji (SDI). W szczególności bazy takie, jak: INSPEC, PASCAL, COMPENDEX, ASSISTENT (CHEMIA, BIOLOGIA, AUTOMATYKA i RADIOELEKTRONIKA, BUDOWA MASZYN, METALURGIA I SPAWALNICTWO), INIS, ISMEC. Istotnym ograniczeniem w rozwoju tej usługi jest to, że w poszczególnych ośrodkach obliczeniowych zainstalowane są zbyt małe pamięci dyskowe.

Ocenia się, że udostępnianie wszystkich sekcji bazy INSPEC z okresu 5 lat wymagałoby przeznaczenia dla systemu IDOL dysków o pojemności kilku gigabajtów. Obecnie w Ośrodku Obliczeniowym Politechniki Wrocławskiej, dysponującym największą pamięcią dyskową, znajduje się tylko 160 MB tej pamięci. Z tego powodu eksploatacja systemu jest możliwa jedynie na małą skalę, w trybie eksperymentalnym. Aktualnie zbiory systemu obejmują tylko jedną sekcję C (komputery i sterowanie) baz danych INSPEC z roku 1983 i 1984.

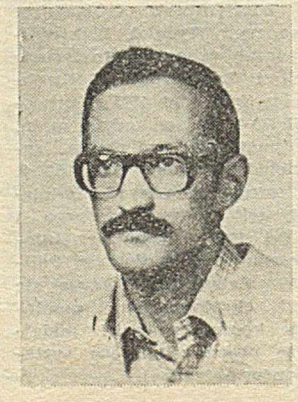
Poszczególne ośrodki podjęły już działania mające na celu zwiększenie pojemności pamięci dyskowych. Istnieje możliwość rozproszonego eksploatowania bibliograficznych baz danych. W takim wariancie poszczególne bazy byłyby udostępniane przez różne ośrodki.

ZASOBY PROGRAMOWE I INFORMOWANIE O ZASOBACH

Inną usługą MSK jest udostępnianie zasobów programowych zgromadzonych dotychczas w każdym ośrodku. Do zasobów tych należą:
— biblioteki programów i podprogramów numerycznych,
— biblioteki programów i podprogramów specjalistycznych (symulacyjne, optymalizujące, obliczeń projektowych, konstrukcyjnych itp.),
— systemy użytkowe i pakiety programów wspierających dydaktykę,
— różne wersje komiplatorów języków itp.
Oprócz obszarów wspólnych zasoby te mają część obszarów rozłącznych i tym samym stwarzają większe możliwości dla użytkownika.

Korzystanie z tych zasobów ułatwiają odpowiednie systemy informowania o nich, opracowane w każdym ośrodku i dostępne z terminala za pomocą mikropoleceń. Użytkownik wydając odpowiednie makropolecenia uzyskuje informacje o sposobie korzystania z systemu informowania, ogólne informacje o udostępnianych zasobach i na życzenie — informacje bardziej szczegółowe dotyczące wybranego zasobu.

Mgr inż. JÓZEF JANYSZEK w 1970 r. ukończył studia na Wydziale Elektroniki Politechniki Wrocławskiej. W latach 1970—1973 pracował w Zakładzie Elektronicznej Techniki Obliczeniowej Budownictwa ETOB w Katowicach na stanowisku inż. elektronika, zajmując się obsługą techniczną m. c. Mińsk 32. Od 1973 r. jest pracownikiem Politechniki Wrocławskiej. Od 1984 r. pełni funkcję dyrektora Ośrodka Obliczeniowego Politechniki Wrocławskiej i administratora Międzyuczelnianej Sieci Komputerowej.



Mgr inż. ZBIGNIEW ZAJĄCZKOWSKI w 1973 r. ukończył studia na Wydziale Elektroniki Politechniki Wrocławskiej. Od tego roku jest zatrudniony w Bibliotece Głównej i OINT Politechniki Wrocławskiej w oddziale Automatyzacji Przetwarzania Informacji Naukowej (APIN). Obecnie pełni funkcję kierownika tego oddziału. Zajmuje się systemami wyszukiwania informacji oraz zautomatyzowanymi systemami biblioteczno-informacyjnymi.

Opracowano również specjalne makropolecenie INFMSK, które umożliwia wyprowadzenie informacji o sieci MSK i jej usługach oraz instrukcji korzystania z języka procesora czołowego, systemu IDOL, systemów informowania itp.

URUCHOMIENIE I WDROŻENIE SIECI MSK

Przez uruchomienie i wdrożenie rozumie się złożony proces, w którym następuje połączenie elementów sprzętowych, programowych i obsługi (operatorzy, administratorzy itp.) w jeden spójny system, udostępniający swe usługi użytkownikom.

Właściwy proces uruchamiania i wdrażania MSK (poprzedzony testami laboratoryjnymi) rozpoczął się w grudniu 1983 r., kiedy rozpoczęto zestawianie linii dzierżawionych i łączących ośrodki w Warszawie i Wrocławiu oraz w Gliwicach i Wrocławiu. Przyjęto przy tym następujące zasady prowadzenia prac:

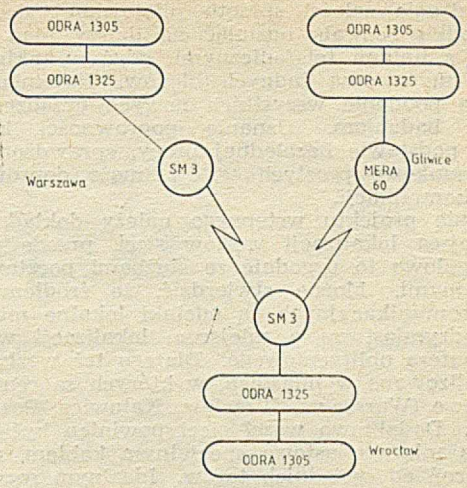
- w pierwszej kolejności uruchomi się i wdroży konfigurację podstawową, w której węzły (dwa na SM-3) będą umiejscowione we Wrocławiu. Konfigurację tę będzie się stopniowo rozwijać dodając kolejno nowe elementy, po sprawdzeniu działania elementów już włączonych;

- eksploatacja próbna tak powstającej sieci będzie prowadzona w sesjach (dwie, trzy sesje tygodniowo), tak aby w jak najmniejszym stopniu zakłócać podstawową pracę lokalną ośrodków. Wyróżniono przy tym trzy rodzaje sesji:
  — sesje testujące wprowadzony nowy element
  — sesje badawcze przeznaczone do przeprowadzania określonych eksperymentów, pomiarów,
  — sesje użytkowe, w których wdrożone zasoby będą udostępnione użytkownikom;

- równolegle będą prowadzone szkolenia odpowiednich służb poszczególnych ośrodków, a także akcja informacyjno-szkoleniowa dla użytkowników.

Postępując zgodnie z przyjętymi zasadami w pierwszej kolejności zestawiono i uruchomiono (grudzień 1983 — styczeń 1984) konfigurację podstawową z dwoma węzłami na komputerach SM-3 we Wrocławiu. Następnie uruchamiano i kolejno włączano węzeł na komputerze SM-3 w Warszawie (styczeń 1984), nową wersję (heterogeniczną) oprogramowania komputera czołowego na komputerze ODRA 1325 (kwiecień 1984), węzeł na komputerze MERA 60 w Gliwicach (maj 1984). W ten sposób powstała eksperymentalna konfiguracja, którą przedstawiono na rys. 1.

Jednym z mankamentów tej konfiguracji jest brak bezpośredniego połączenia węzłów Warszawa — Gliwice (trudna sytuacja z łączami w okolicach Gliwic), co uniemożliwia testowanie algorytmów trasowania i obniża sprawność podsieci.



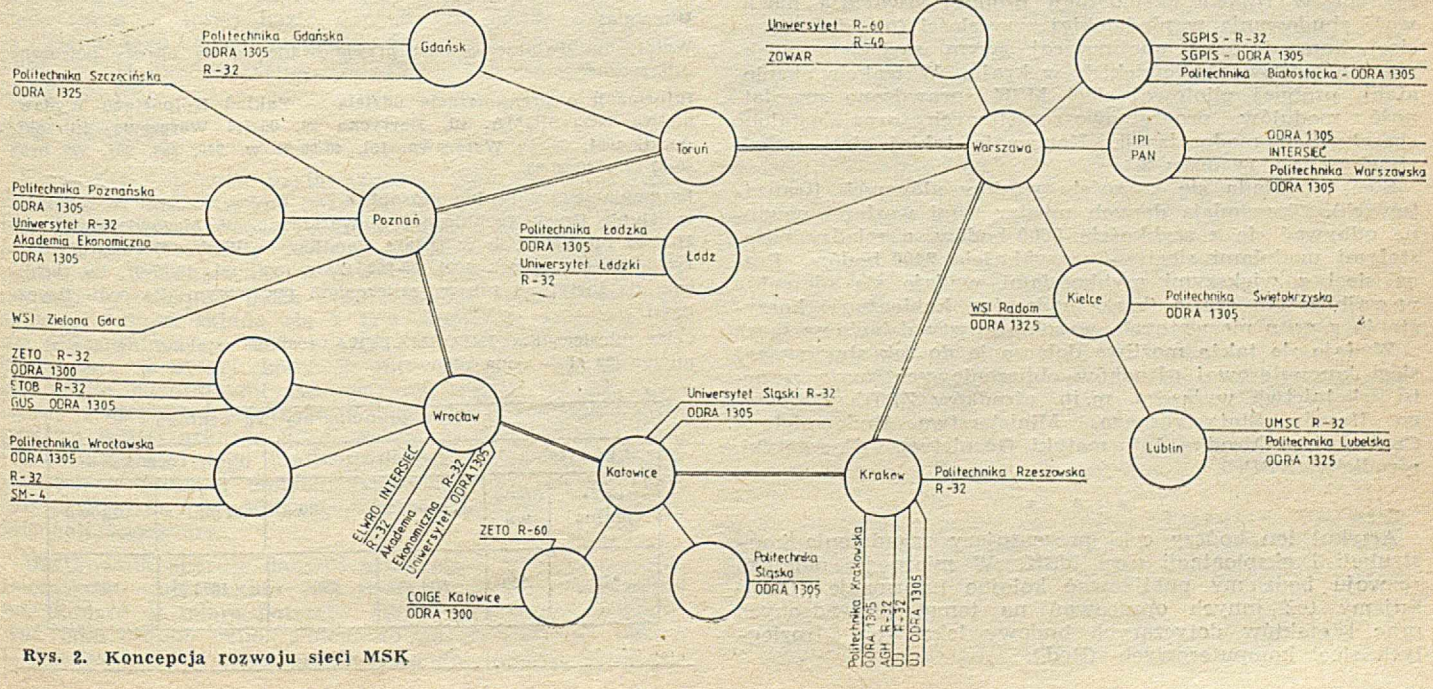
Rys. 1. Eksperymentalna konfiguracja pilotowej sieci MSK (maj 1984 — marzec 1985)

Równolegle z rozbudową konfiguracji prowadzono prace polegające na wdrożeniu systemu IDOL, lokalnych systemów informowania, procedur organizacyjno-technologicznych eksploatacji systemu pomiarowo-diagnostycznego podsieci. Do uruchomienia i wdrożenia pozostały jeszcze: system transferu zbiorów i zadań dla komputera obliczeniowego ODRA 1305, system transferu zbiorów dla komputera obliczeniowego R-32, komputer czołowy na EC 8371-01 dla komputera obliczeniowego R-32, koncentrator terminali na komputerze SM-4.

DOŚWIADCZENIA UZYSKANE W TRAKCIE URUCHAMIANIA MSK

Uruchomienie MSK było pierwszym tego rodzaju przedsięwzięciem w Polsce. Brak więc było jakichkolwiek doświadczeń i wzorów. Przyjmując nawet pewne wzorce z uruchomienia sieci poza granicami kraju, należało uwzględnić pewną specyfikę krajową, na przykład bardzo duże opóźnienie w rozwoju sieci telefonicznej.

Uruchomienie sieci należy podzielić na dwa etapy: etap uruchomienia w warunkach laboratoryjnych i etap uruchomienia właściwego, tj. w warunkach rzeczywistych. Uruchomienie sieci w warunkach laboratoryjnych umożliwia obserwację pulpitów technicznych poszczególnych modułów oraz korzystanie z pułapek programowych i sprzętowych, wnoszenie poprawek do systemów operacyjnych z klawiatur, natychmiastowe wykonanie wydruków systemów operacyjnych w wypadku ich załamania.



Rys. 2. Koncepcja rozwoju sieci MSK

Zgromadzenie całości sprzętu pilotowanej sieci w jednym ośrodku pozwala odsunąć na dalszy plan problemy związane z liniami telefonicznymi, gdyż w takim wypadku linie nie wnoszą żadnych błędów. Natomiast wydaje się ważne poddanie wszystkich nowych urządzeń wszechstronnym badaniom. Uznanie poprawności konstrukcji tylko na podstawie bezbłędnej pracy, sprawdzonej za pomocą stosunkowo prostych testów, może doprowadzić do sporych komplikacji.

Na etapie projektu wstępnego należy dokładnie rozpatrzyć miejsca lokalizacji węzłów sieci, przede wszystkim zaś szczegółowo to uzgodnić ze służbami pocztowo-telekomunikacyjnymi. Można stwierdzić, że źródłem zakłóceń linii telekomunikacyjnych są odcinki lokalne między centralą międzymiastową a miejscem lokalizacji węzła sieci lub komputera obliczeniowego. Dlatego też węzły powinny być zlokalizowane w miastach, w których są centrale międzymiastowe (Warszawa, Wrocław, Katowice, Poznań, Kraków itp.). Dodatkowo węzeł sieci powinien być połączony z centralą międzymiastową specjalnym kablem, aby uniknąć zakłóceń od sąsiednich torów. Idealnym rozwiązaniem byłoby, aby węzły sieci stanowiły wyposażenie central międzymiastowych.

Przy uruchomieniu sieci komputerowych w polskich warunkach nie należy korzystać z łącz komutowanych z powodu:
— trudności wynikających ze składania łącz komutowanych dwutorowych (każdy tor należy składać oddzielnie),
— częstego rozłączania złożonych łącz,
— konieczności pracy na wysokim poziomie nadawczym, co prowadzi do zakłóceń w torach sąsiednich.

W projekcie techniczno-funkcjonalnym MSK przyjęto założenie, że powinna być ona zbudowana z elementów produkcji krajowej lub krajów RWPG. W konsekwencji powstało wiele elementów sieci, np. adaptery ALS11, ALS11M, UAL01, wykonanych w nie najnowszej technologii. Niewielki import elementów mikroprocesorowych z krajów zachodnich, np. sterowników SDLC/HDLC 8273 i innych, mógłby w znacznym stopniu zmniejszyć wydatki poniesione na zbudowanie pilotowej sieci MSK.

Projektanci narzucili również sztywne sposoby dołączania komputerów obliczeniowych typu RIAD i ODRA do sieci. Wielu potencjalnych użytkowników sieci nie może sprostać tym wymaganiom z powodu trudności uzyskania adaptera ADM 305, komputera ODRA 1325, teleprocesora EC 8371, adapterów ALS11, UAL01 itp. Dlatego też zachodzi potrzeba opracowania wariantowych rozwiązań, nawet gorszych z punktu widzenia obciążenia komputera obliczeniowego, ale możliwych do zaakceptowania przez przyszłego użytkownika. Przy dołączaniu komputera obliczeniowego do sieci powinna być przestrzegana zasada, że włączenie to nie może ograniczać dotychczasowych jego aplikacji. W wypadku komputera ODRA zasada ta jest spełniona, a dla komputera R-32 odstąpiono od tej zasady.

* * *

Uruchomienie pilotowej sieci MSK i doświadczenia uzyskane w trakcie eksploatacji próbnej stwarzają możliwość zbudowania w niedalekiej przyszłości (np. do 1990 r.) sieci komputerowej obejmującej swym zasięgiem większość ośrodków akademickich w kraju. W trakcie eksploatacji próbnej pilotowej sieci MSK sprawdzono przydatność modułów programowo-sprzętowych oraz ustalono sposoby ich modernizacji. Koncepcję takiej sieci przedstawiono na rysunku 2.

Sieć ta składa się z sześciu węzłów głównych (tranzytowych). Transmisja danych między tymi węzłami powinna odbywać się z szybkością 9600 bodów, a między pozostałymi modułami sieci — z szybkością 2400 bodów. Praca sieci z większymi szybkościami wydaje się nierealna w najbliższym czasie (brak modemów, konieczność korzystania z grup pierwotnych, wysokie koszty dzierżawy łączy).

Wydaje się także możliwe dołączenie do tak zbudowanej sieci komputerowej ośrodków obliczeniowych spoza resortu szkolnictwa wyższego, m. in. ośrodków ZETO, Głównego Urzędu Statystycznego, Ministerstwa Budownictwa, Centralnego Ośrodka Informatyki Górnictwa i Energetyki, ośrodków NBP.

* * *

Artykuł ten kończy cykl prezentujący zagadnienia konstrukcji i eksploatacji sieci MSK. W miarę jej dalszego rozwoju będziemy publikować kolejne informacje. Oczekujemy też innych opracowań na temat prowadzonych prac i efektów dotyczących budowy lokalnych i rozległych sieci komputerowych (Red.).

LIERATURA

[1] Bieleninik E., Hudyma E., Kosmulska-Bochenek E., Lewoc J., Stanisz A.: Podsieć komunikacyjna Międzyuczelnianej Sieci Komputerowej, Informatyka, nr 11—12, 1985

[2] Bordnik M., Mikołajewski S., Poręba J., Siwek Z.: Budowa i badania jakości sieci teledacji w sieci pilotowej MSK, Raport Instytutu Telekomunikacji i Akustyki Politechniki Wrocławskiej, nr SPR-014, 1984

[3] Budzianowski L., Kwiatkowski J., Wietrzych J.: Komputer ODRA 1305 w Międzyuczelnianej Sieci Komputerowej, Informatyka, nr 7—8, 1986

[4] Czajczyńska E., Góral S., Kupczyk M., Matuszewska-Gruber B.: Projekt techniczno-funkcjonalny sieci MSK. Cz. V. Technologia eksploatacji zasobów technicznych pilotowej sieci MSK, Raport Centrum Obliczeniowego Politechniki Wrocławskiej, nr 18, 1983

[5] Janczewski K.: Badania eksperymentalne i pomiary pilotowej sieci MSK. Raport Centrum Obliczeniowego Politechniki Wrocławskiej, nr 44, 1984

[6] Janyszek J.: Uruchomienie i pomiary MSK. Raport Centrum Obliczeniowego Politechniki Wrocławskiej, nr 13, 1984

[7] Pietrzak A., Statkiewicz I., Zajączkowski Z.: IDOL — system wyszukiwania informacji retrospektywnych na maszynach cyfrowych serii ODRA 1300. Dokumentacja użytkowa. Raport Biblioteki Głównej i OINT Politechniki Wrocławskiej, nr 42, 1982.

WARUNKI PRENUMERATY NA 1987 R.

Prenumeratorzy zbiorowi — jednostki gospodarki uspołecznionej, instytucje i organizacje społeczne zamawiają prenumeratę dokonując wpłat na blankiecie „polecenie przelewu".

Prenumeratorzy indywidualni — osoby fizyczne zamawiają prenumeratę dokonując wpłaty w UPT lub NBP na blankiecie Wydawnictwa lub blankiecie NBP. Na odwrocie wszystkich odcinków blankietu należy wpisać tytuł czasopisma, okres prenumeraty, liczbę zamawianych egzemplarzy oraz wartość wpłaty.

Wpłacać należy na konto NBP III O/M Warszawa 1036-7490-139-11.

Prenumerata ulgowa — przysługuje wyłącznie osobom fizycznym — członkom SNT, studentom i uczniom szkół zawodowych. Warunkiem prenumeraty ulgowej jest poświadczenie blankietu wpłaty (przed jej dokonaniem) na wszystkich odcinkach pieczęcią Koła SNT, wyższej uczelni lub szkoły.

Sposób zamawiania prenumeraty taki sam jak dla prenumeraty indywidualnej.

Prenumerata ze zleceniem wysyłki za granicę — zamawia się tak jak prenumeratę indywidualną. Dodatkowo należy podać na blankiecie wpłaty nazwisko i dokładny adres odbiorcy. Cena prenumeraty ze zleceniem wysyłki za granicę jest dwukrotnie wyższa.

Przedpłaty na prenumeratę przyjmowane są w terminach:
— do 10 listopada na I kwartał, I półrocze i cały rok następny,
— do 28 lutego na II, III, IV kwartał i II półrocze,
— do 31 maja na III, IV kwartał i II półrocze,
— do 31 sierpnia na IV kwartał.

Uwaga!

Wpłaty na dwumiesięczniki przyjmowane są na okresy półroczne lub roczne.

Informacji o prenumeracie udziela — Zakład Kolportażu Wydawnictwa NOT-SIGMA, ul. Bartycka 20, 00-716 Warszawa, lub skr. poczt. 1004, 00-950 Warszawa, tel. 40-00-21 w. 249, 293, 297, 299 oraz 40-35-89 i 40-30-86.

Egzemplarze archiwalne czasopism — można nabyć za gotówkę w Klubie Prasy Technicznej w Warszawie, ul. Mazowiecka 12, tel. 27-43-65 lub zamówić w Dziale Handlowym Wydawnictwa, ul. Bartycka 20, skr. poczt. 1004, 00-950 Warszawa, tel. 40-37-31, na rachunek dla instytucji lub za zaliczeniem pocztowym dla osób fizycznych.

Cena miesięcznika INFORMATYKA została ustalona na 150 zł za numer (50 zł — cena ulgowa).

Cena prenumeraty wg cennika					
kwartalna		półroczna		roczna	
normalna	ulgowa	normalna	ulgowa	normalna	ulgowa
450	150	900	300	1800	600

GRZEGORZ TRAWIŃSKI
PPZ Computex
Warszawa

Pakiet zintegrowany COMPUTEX

Rozwój techniki komputerowej i mikrokomputerowej doprowadził do sytuacji, w której rzeczą drugorzędną staje się posiadanie komputera jako urządzenia, natomiast zasadniczego znaczenia nabiera jego oprogramowanie. Wymaga się, aby było ono uniwersalne, łatwe w obsłudze i odporne na błędy operatora, sprzętu itp. Poza tradycyjnym oprogramowaniem podstawowym (np. kompilatory, edytory itd.), coraz popularniejsze są programy realizujące konkretne zadania na poziomie użytkowym — graficzne, kalkulacyjne, operujące bazami danych, umożliwiające projektowanie i in. Coraz trudniej jest nauczyć się używania wszystkich programów, z których każdy ma zazwyczaj inne instrukcje, inne formaty danych (co utrudnia lub uniemożliwa przenoszenie wyników do innych programów), inny sposób komunikacji z użytkownikiem. W takiej sytuacji, aby ułatwić pracę użytkownikom, zaczęto rozwijać nową generację oprogramowania — pakiety programowe łączące w sobie kilka funkcji, tzw. pakiety zintegrowane.

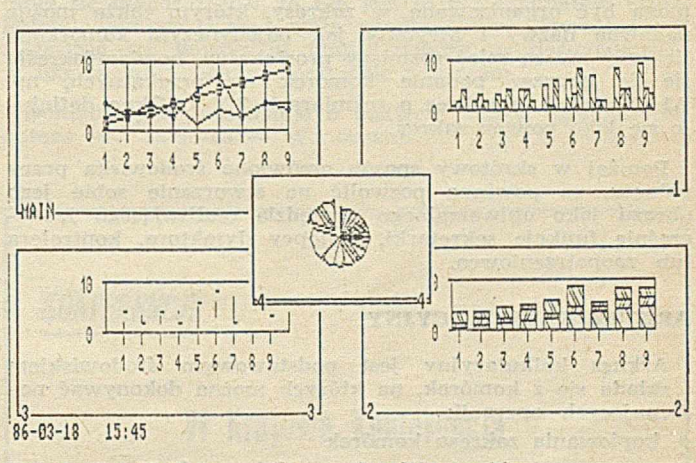
W niniejszym artykule przedstawiono podstawowe informacje o pakiecie programowym Computex, należącym do nowego jakościowo rodzaju oprogramowania dla mikrokomputerów osobistych zgodnych z IBM PC.

Dotychczas istniejące programy użytkowe operowały zazwyczaj jednym, określonym rodzajem danych, np. programy typu dBASE — bazami danych, VisiCalc — arkuszami kalkulacyjnymi itd. Computex zrywa z tą tradycją, umożliwiając pracę w różnych środowiskach na tych samych zestawach danych (przez środowisko należy rozumieć sposób interpretacji danych przez programy pakietu). Przykładowo, informacje typu ekonomicznego o pracy przedsiębiorstwa — po wprowadzeniu do bazy danych — mogą być zilustrowane różnymi wykresami, w sposób przejrzysty ujawniającymi bieżącą sytuację i kierunki rozwoju; tekst oferty lub korespondencji w łatwy sposób można skierować do odbiorców, o których dane są przechowywane w innej bazie danych (ale także istniejącej w obrębie tego samego pakietu); do wymienionych baz danych można automatycznie włączyć dane z arkusza kalkulacyjnego itd.

OPIS OGÓLNY

Computex zapewnia użytkownikowi pięć środowisk pracy: arkusze kalkulacyjne, bazy danych, przetwarzanie tekstów, komunikację między komputerami i wykresy ilustrujące dane. Wszystkie informacje przechowywane przez pakiet są zorganizowane w arkusze — w danej chwili w pamięci operacyjnej może znajdować się tylko jeden taki arkusz. Arkusz składa się z tzw. komórek, podzielonych na 256 kolumn w 8192 wierszach. Komórki mogą być grupowane w zbiory komórek (tzw. zakresy) charakteryzujące się określonymi parametrami, można im nadawać nazwy i atrybuty. Użytkownik decyduje o sposobie interpretacji całego arkusza lub jego części, tworząc tzw. okno i określając dla niego środowisko pracy. Taka organizacja pamięci, a zatem i danych, pozwala przez zmianę okna lub środowiska na odmienne traktowanie tych samych informacji z arkusza.

Użytkownik operuje danymi przez jedno lub kilka okien. Początkowo cały arkusz jest dostępny przez okno główne, dopiero w miarę potrzeby można jego część zdefiniować jako osobne okno. Polega to na nadaniu mu nazwy i określeniu, jaką część arkusza ma obejmować, jakie będzie jego środowisko pracy, jaki obszar będzie zajmować na ekranie. Podaje się też inne mniej istotne parametry. Okno główne obejmuje cały arkusz i powierzchnię całego ekranu. Wszystkie parametry okna mogą być zmienione w czasie pracy. Kilka okien może mieć wspólne części arkusza i ekranu (w tym ostatnim wypadku okna nieaktywne będą przesłonięte przez okno bieżące i widoczne tylko wtedy, gdy są umieszczone w innej części ekranu lub mają większe rozmiary). Przykładowe rozmieszczenie różnych okien przedstawiono na rys. 1.



Rys. 1. Przykładowe rozmieszczenie różnych okien

Każda komórka arkusza, oprócz nazwy wynikającej z pozycji w arkuszu — np. C12 oznacza dwunastą komórkę w trzecim wierszu — może mieć nadaną nazwę, np. NAZWA-TOWARU. Komórka może przechowywać daną jako liczbę, napis (tzw. etykietę), wyrażenie numeryczne lub napisowe (tzw. formułę).

Liczba, tj. dana numeryczna z przedziału (9.9e-99, 9.9e99), może być przechowywana w jednym z ośmiu standardów określających tzw. sposób interpunkcji i w jedenastu formatach określających sposób interpretacji i wyświetlania. W tym drugim wypadku wyróżnia się:

- format waluty, w którym do liczby dodaje się znak waluty (określany przez użytkownika) jako przedrostek lub przyrostek ze stałą liczbą cyfr znaczących (od 0 do 15), np. $ 33.43; 32.12 $; 345,50 zł; wartości ujemne ujęte są w nawiasy, np. (4000,00 zł).;
- format zmiennopozycyjny, zgodny z aktualnym sposobem interpunkcji, np. 123.456,78901;
- format stałopozycyjny z określoną liczbą miejsc dziesiętnych (od 0 do 15);
- procenty, np. 3,6% zgodnie z aktualnymi wartościami parametrów;
- format ogólny, w którym bardzo duże i małe liczby są wyświetlane w postaci wykładniczej, np. 123,43; -6; 123434545.454; 1.23e-33;
- daty w formie pełnej lub krótkiej (bez roku), tzn. MM/DD/RR, DD/MM/RR, DD.MM.RR, RR-MM-DD, DD-MMM-RR, DD-MMM, MMM-RR, np. 15-Maj-86, 15/05/86, 15/05, 86-05-15 itd.;
- czas w formie pełnej lub krótkiej (bez sekund), tzn. HH:MM:SS, HH.MM.SS, HH,MM,SS, HHhMMmSSs, HH:

:MM:SS AM/PM, np. 11:24:43, 13:13, 06h34m13s, 11:23 AM itd.;

- format naukowy z określoną liczbą miejsc mantysy (od 0 do 15), np. 1.234e-33, 0,3245467e+00 itd.;
- symboliczny histogram poziomy, w którym liczba znaków symbolizuje wartość dodatnią (+), ujemną (—) lub zero (.), np. ++++ (liczba +4);
- komórkę ukrytą, której wartość nie jest wyświetlana (możliwości ochrony arkusza omówiono poniżej);
- postaci formuły zamiast wartości (p. poniżej).

Etykieta jest ciągiem do 240 znaków ASCII traktowanym jak napis.

Formuła jest to wyrażenie zależne od dowolnej liczby innych komórek lub wartości. W skład formuły mogą wchodzić liczby, napisy, adresy komórek, nazwy zakresów, tzw. ▫ — funkcje (wbudowane funkcje standardowe pakietu). Można używać wszystkich operatorów arytmetycznych, logicznych i napisowych. Ponieważ formuła może zależeć od zawartości innych komórek (które także mogą zawierać formuły), przy zmianie jednej wartości w komórce modyfikacji ulegną wszystkie komórki z nią powiązane.

Komórka może mieć takie atrybuty, jak format numeryczny, szerokość na ekranie i sposób ochrony. Komórki mogą być organizowane w zakresy, którym także można nadawać nazwy i atrybuty jak pojedynczym komórkom. Zakresy muszą mieć rozmiary prostokątne, bowiem określa się je poprzez podanie komórek na przekątnych, np. A1..C7 definiuje zakres o rozmiarach 7 × 3. Okno definiuje się jako pewien zakres.

Poniżej w skrótowy sposób omówiono środowiska pracy pakietu, co powinno pozwolić na stworzenie sobie jego obrazu jako uniwersalnego narzędzia realizującego jednocześnie funkcję sekretarki, zastępcy dyrektora, kontrolera lub zaopatrzeniowca.

ARKUSZ KALKULACYJNY

Arkusz kalkulacyjny jest podstawowym środowiskiem i składa się z komórek, na których można dokonywać następujących operacji:

- kopiowania zakresu komórek
- przesuwania komórek w inne miejsce arkusza
- usuwania ich
- ustalania lub zmieniania formatu numerycznego
- nazywania zakresów
- zamiany wierszy z kolumnami
- zamiany formuł na wartości
- zmiany sposobu wyrównania etykiet
- wypełnienia zakresu wartościami
- tworzenia tablic decyzyjnych
- ochrony polegającej na zamknięciu dostępu do zawartości komórki (zakaz wyświetlania lub dokonywania zmian); aby dokonać jakiejkolwiek operacji na takiej komórce, należy podać wcześniej ustalone hasło.

Ponadto w tym środowisku można dokonywać operacji na bazach danych i używać wykresów.

BAZY DANYCH

Środowisko baz danych pozwala na administrowanie zestawami informacji. Definiowanie bazy danych przez użytkownika jest bardzo proste i polega na podaniu zakresu zawierającego etykiety, które staną się selektorami pól rekordów bazy danych. Każde pole ma wartość domyślną, wstawianą w wypadku, gdy nie wprowadzi się żadnej wartości. Można podać formułę odrębna dla każdego pola, według której zostanie ono przekształcone i umieszczone w bazie, np. można przekształcić wprowadzony napis w polu NAZWISKO tak, aby pierwsza litera była wielka, a pozostałe małe, dzięki czemu wpisanie słowa kowalski, KOWalsKi itp. zawsze spowoduje umieszczenie napisu Kowalski. Dla każdego pola można określić warunek logiczny, zezwalający pakietowi na weryfikację poprawności wprowadzanej danej, np. formuła +CENA >= 400 spowoduje, że jeżeli wprowadzona wartość pola CENA będzie mniejsza od 400, to nie zostanie ona umieszczona w bazie danych (przyjmuje się wartość domyślną — np. 0).

W bazie danych można dokonywać m. in. następujących operacji:

- sortowania według trzech podanych kluczy; porządkowanie może obejmować rekordy identyczne (będą wtedy umieszczone obok siebie) lub eliminować powtarzające się, zastępując je pojedynczym;
- przywracania wybranym rekordom wartości początkowych (inicjalizacja);
- selektywnego działania na rekordach spełniających podane kryteria (np. w bazie danych zawierających listę pracowników może być potrzebne wybranie tylko tych, którzy pracują więcej niż 10 lat);
- tworzenie raportów, tzn. drukowania zawartości bazy danych lub wybranych informacji w określonej formie (np. można przygotować wzór pisma do klientów, pozostawiając miejsce na nazwisko, imię, adres, rodzaj towaru itd. i zażądać wydrukowania go dla każdego indywidualnie, ze wstawieniem potrzebnych wartości z bazy danych).

Bazami danych można również operować w środowisku arkuszy kalkulacyjnych, przy czym dodatkowo istnieje możliwość kopiowania części bazy danych spełniających podane kryteria na inny obszar arkusza. Do bazy danych można włączać informacje powstałe na skutek analizy dowolnego tekstu.

PRZETWARZANIE TEKSTÓW

Praca w środowisku przetwarzania tekstów umożliwia tworzenie, poprawianie i formatowanie dowolnych dokumentów. Użytkownik może tworzyć specjalne linie, tzw. linie formatu, zawierające wskazówki dla pakietu, w jaki sposób ma być formatowany tekst poniżej tej linii aż do następnej linii formatu. Można określić odstępy tabulacji, sposób adiustacji tekstu, wielkość lewego i prawego marginesu, wyświetlanie odstępów w sposób widoczny, wyświetlanie znaków końca akapitu, automatyczną adiustację bądź jej brak. Ponadto istnieje możliwość tworzenia lokalnych wcięć wewnątrz akapitu tekstu.

Zestaw operacji na tekstach pozwala m. in. na: kopiowanie, przesuwanie, usuwanie dowolnych fragmentów tekstu, wyszukiwanie i zastępowanie kontekstowe, adiustację całości lub części tekstu, omówione operowanie liniami formatu, stronicowanie tekstu.

Operowanie tekstami jest szczególnie łatwe dzięki możliwości poruszania się po całych słowach, zdaniach i akapitach. Ponadto można określić atrybuty wybranych fragmentów, np. wytłuszczenie, pisanie kursywą, z podkreślaniem itd.

KOMUNIKACJA

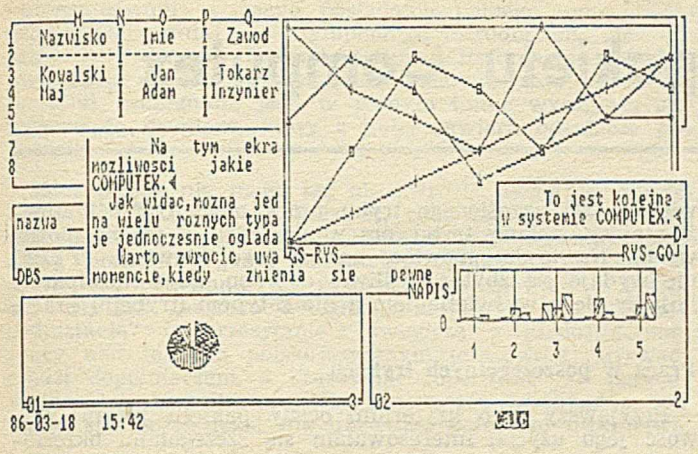
Środowisko komunikacji w pakiecie zostało przewidziane jako rozwiązanie do wykorzystania w przyszłości, ponieważ obecnie istnieją zbyt liczne ograniczenia sprzętowe i formalne. Pakiet Computex pozwala użytkownikowi zaopatrzonemu w modem i telefon komunikować się z innymi komputerami lub emulować terminal oddalonego komputera (np. SM-4). Przy połączeniu przez linię telefoniczną Computex sam potrafi wybrać numer, wysłać i odebrać wiadomości oraz dane, zapisując je do arkusza, na dysk lub drukarkę. Nawet podczas pracy w innym oknie, o ile istnieje okno komunikacji, pakiet może odbierać dane i poinformować użytkownika o stanie połączenia. Przy emulacji terminala możliwe jest automatyczne podanie konta i hasła oraz poleceń dla systemu operacyjnego.

TWORZENIE WYKRESÓW

Środowisko wykresów jest jedynym, które zależy od danych istniejących w innych trybach pracy. W pewnym sensie pełni ono rolę służebną, zezwalając użytkownikowi na ilustrowanie danych z dowolnego okna arkusza na monitorze, drukarce, a także na ploterze za pomocą następujących rodzajów wykresów (rys. 2): histogramów w dwóch postaciach, wykresów we współrzędnych prostokątnych w dwóch postaciach, wykresów kołowych, wykresów tendencji.

Wykresy mogą być skalowane (liniowo lub logarytmicznie), opisywane na osiach, przy punktach danych i za pomocą tekstu objaśniającego. Można wybierać poziom odniesienia, siatkę nanoszoną na obraz, kształt wykresu ko-

łowego itd. W środowisku arkuszy kalkulacyjnych za pomocą pojedynczej komendy można ilustrować dane w taki sam sposób jak w środowisku wykresów.



Rys. 2. Różne rodzaje wykresów w pakiecie

Użytkownicy nie dysponujący drukarkami lub ploterami produkcji zagranicznej mogą także korzystać ze wszystkich opisanych możliwości. Dołączone do niniejszego artykułu rysunki zostały wykonane na polskiej drukarce D-100 sterowanej za pomocą opracowanego przez PPZ COMPUTEX programu GRAFIK, który może współpracować z omawianym pakietem.

POZOSTAŁE WŁAŚCIWOŚCI

Poza pracą w oknach, COMPUTEX administruje systemem plików, umożliwia drukowanie danych według podanego wzoru (z nagłówkiem, stopką, marginesami, tytułami stron itd.), zapewnia utajnienie danych, wykonanie poleceń systemu operacyjnego bez przerywania sesji oraz inne ułatwienia.

W każdym środowisku istnieje wiele wbudowanych funkcji (tzw. D — funkcji) arytmetycznych, logicznych, napisowych, statystycznych oraz użytkowych, które zezwalają na efektywniejsze działanie na danych np. D SIN — funkcja sinus, D SUM — suma wartości w zakresie, D RAND — liczba pseudolosowa, D VALUE — zamiana znaków ASCII na liczbę, D STD — odchylenie standardowe itd.

Szczególnie cenną cechą pakietu jest możliwość pisania programów w jego własnym języku, zbliżonym do Basica lub Fortranu, dzięki czemu dowolne sekwencje operacji mogą być łatwo powtarzane lub nawet wykonywane podczas nieobecności użytkownika (ta cecha może być najlepiej wykorzystana wraz z możliwościami komunikacyjnymi — pakiet może pełnić rolę elektronicznej sekretarki). W szczególności pakiet może o żądanej porze przypomnieć o zaplanowanej wcześniej czynności — realizuje to poniższy program (symbole ujęte w nawiasy klamrowe są komendami języka):

```
{WAIT D timevalue("15:15")-D now}
                    oczekuj do godz. 15:15;
                    D timevalue jest funkcją udostępniającą liczbę odpowiadającą podanemu czasowi, a
                    D now udostępnia bieżący czas;
{BEEP}              alarm dźwiękowy;
{PROMPT „Już czas na
  konferencję"}     komunikat;
```

Język umożliwa m. in. tworzenie pętli FOR, pozwala na korzystanie z podprogramów wraz z parametrami, sprawdzając ich formalną zgodność z deklaracją, umożliwia krokowe wykonanie programu, co ułatwia jego sprawdzenie i uruchomienie.

Computex komunikuje się z użytkownikiem za pośrednictwem menu, jednocześnie krótko opisując wybraną pozycję, co umożliwia pracę nawet takiemu użytkownikowi, który po raz pierwszy zetknął się z tym pakietem. Operacje mogące spowodować utratę danych (np. skasowanie pliku lub zakończenie pracy bez zapisania danych) wymagają potwierdzenia, co zmniejsza możliwość popełnienia błędu. Dodatkowo w każdej chwili, nie przerywając pracy, można skorzystać z elektronicznego podręcznika zawierającego wszelkie pomocne informacje. Pakiet śledzi wykonywane czynności, tak że podręcznik uaktywniany jest w miejscu opisującym bieżącą sytuację.

Ważną cechą pakietu jest zdolność do włączania do własnych danych wyników pracy innych programów, np. edytorów, a także programów specjalizowanych, jak dBASE, VisiCalc itd., jak również — możliwość tworzenia danych w formatach odpowiadających tym programom, co zapewnia wymienialność i ciągłość pracy nawet przy zmianie narzędzia.

* * *

Pełna dokumentacja pakietu Computex obejmuje dwa podręczniki formatu A4, po ok. 400 stron każdy. Nie oznacza to jednak, że należy ją natychmiast szczegółowo przestudiować; powinno to następować stopniowo i selektywnie w miarę potrzeby. Najlepiej jest uruchomić pakiet i zacząć pracę, pozwalając prowadzić się przez wyświetlane menu i korzystając z elektronicznego podręcznika. Dopiero potem można sięgnąć do dokumentacji i zobaczyć, jak w efektywniejszy sposób zrobić to samo w przyszłości. Niewiele byłoby przesady w stwierdzeniu, że jeżeli nie umie się czegoś zrobić, to nie jest to niedoskonałość Computexu, lecz wciąż zbyt mała wiedza o nim. Pracę można rozpocząć na dowolnym etapie wiedzy, a doświadczenie w użytkowaniu pakietu przyjdzie z czasem.

Dokładniejsze informacje o pakiecie można otrzymać od autora (tel. 23-94-03 — Warszawa).

KONFERENCJE

VI Krajowa Konferencja „Polioptymalizacja w projektowaniu"

Wyższa Szkoła Inżynierska w Koszalinie organizuje w dniach 8—13 czerwca 1987 r. w Ośrodku Szkoleniowo-Wypoczynkowym w Mielnie, kolejną konferencję pt. „Polioptymalizacja w projektowaniu".

Konferencja poświęcona jest problemom optymalizacji i polioptymalizacji (teorii, metodom i zastosowaniom) w projektowaniu technicznym, a także w procesie podejmowania decyzji w sterowaniu procesami technologicznymi, w zarządzaniu i wszędzie tam, gdzie podejmuje się decyzje na podstawie wielu konfliktowych kryteriów (np. w zadaniach ekonomicznych, medycznych, transportowych, organizacyjnych, wojskowych itp.).

Referaty będą opublikowane przed konferencją w specjalnym Zeszycie Naukowym WSI. Przewiduje się ok. 20-minutowy czas wystąpienia. Podczas konferencji do dyspozycji uczestników będzie sprzęt mikrokomputerowy, umożliwiający demonstrowanie własnych programów.

Zgłoszenia i zapytania o warunki uczestnictwa prosimy kierować pod adresem:

Barbara Stępień, WSI Koszalin
ul. Racławicka 15/17, 75-620 Koszalin
lub telefonicznie:

Koszalin, tel. 278-81 wewn. 239 (doc. dr inż. Wojciech Tarnowski) lub wewn. 229 (Barbara Stępień).

Orkiestra na pięć fajerek, czyli niezależna ocena pakietu Computex

Tytuł tego artykułu jest aluzją do nazwy pakietu zintegrowanego Symphony, firmy Lotus, którego dokładnym odpowiednikiem jest Computex. Każdej z tytułowych „fajerek" odpowiada jeden z pięciu trybów pracy Symphony i Computexa: arkusz kalkulacyjny, bazy danych, tworzenie wykresów, przetwarzanie tekstów i komunikacja.

Ponieważ artykuł pt. „Pakiet zintegrowany Computex" ma w pewnym stopniu charakter reklamowy, Redakcja postanowiła sprawdzić w praktyce działanie opisywanego pakietu. Nie zanalizowano go jednak pod względem możliwości, gdyż te są opisane skrotowo w wymienionym artykule, a także dość dokładnie w literaturze anglojęzycznej (por. Byte, No. 7, July 1984 oraz PC World, July 1984).

Podstawowym celem analizy nie było też stwierdzenie, jak dalece tego rodzaju pakiet programowy może być przydatny użytkownikom krajowym, także z uwagi na quasi-polskojęzyczne objaśnienia poleceń (tzn. bez znaków diakrytycznych).

W ciągu krótkiego czasu, który miałam do dyspozycji, mogłam sprawdzić jedynie to, jak łatwo jest posługiwać się tym pakietem. Szczególnie zainteresowało mnie stwierdzenie, w którym autor cytowanego artykułu wyjaśnia, że dzięki komunikacji z użytkownikiem za pomocą menu Computex „umożliwia pracę nawet takiemu użytkownikowi, który po raz pierwszy zetknął się z tym pakietem". **Łatwość użycia** była więc jedynym kryterium, ze względu na które starałam się ocenić ten pakiet.

Pierwsze kroki

Pierwszym krokiem była próba zainstalowania pakietu. I tu czekała mnie niespodzianka — pakiet nie ładował się do pamięci, a przynajmniej nie zaobserwowałam żadnych spodziewanych objawów na ekranie monitora. Przyczyna okazała się dość prosta — pakiet udało się zainstalować po przełączeniu monitora na kartę graficzną.

Po kilku naturalnych, przy pierwszym kontakcie, posunięciach „na ślepo" i uzyskaniu niezrozumiałych odpowiedzi, spróbowałam zbadać możliwości pakietu bardziej systematycznie.

Dopiero po kilku (może kilkunastu) minutach poszukiwań i namysłu udało mi się stwierdzić, że polecenie **EXIT**, oznaczające upragnione opuszczenie Computexa, można wykonać w stanie **SERVICES**, do którego należy wejść przyciskając klawisz **F9**. Oczywiście wtedy, gdy opanowałam technikę kończenia pracy z Computexem, nie wydało mi się to skomplikowane. Informacja na ten temat (**F9--SERVICES, EXIT, Yes**) powinna być jednak podana w sposób bardziej oczywisty i czytelny.

Podobną trudność napotkałam przy wykonywaniu jednej z najbardziej potrzebnych operacji, tj. próbie zapisania na dysku nowo utworzonego pliku. Teraz już wiem, że w stanie **SERVICES** (**F9**) należy wybrać opcję **FILE**, a następnie **SAVE**.

Po skutecznym skorzystaniu z tych dwóch usług zrozumiałam, że polecenia **SERVICES** są najbardziej użyteczne, czego jednakże powinnam dowiedzieć się wcześniej, z podręcznika, zanim przystąpiłam do pracy. Tego niestety w 33-stronicowym wprowadzeniu do Computexa nie znalazłam.

Ponieważ w opisie używa się rodzajowych nazw klawiszy, wkrótce stało się również oczywiste, że zawsze pod ręką powinien być skrócony opis klawiszy funkcyjnych **F1—F10**, które wraz z klawiszem **ALT** umożliwiają wywołanie 23 poleceń w którymś z pięciu trybów pracy. Oprócz klawisza **F9**, najważniejszy jest klawisz **F1** wywołujący samouczek (**HELP**), a także **F10** wywołujący **MENU** odpowiednie dla określonego trybu pracy oraz **ALT-F10** umożliwiający zmianę trybu pracy na jeden z pięciu możliwych. Nawiasem mówiąc, nazwa tego klawisza (**TYPE**) nie wydaje się zbyt szczęśliwie dobrana, gdyż nieodparcie kojarzy się z wyświetlaniem a nie z typem (trybem) pracy.

Praca w poszczególnych trybach

Przyjąwszy jako kryterium oceny pakietu jedynie łatwość jego użycia, interesowałam się szczegółami określonych trybów pracy tylko w stopniu niezbędnym do prawidłowego posługiwania się pakietem.

Najbardziej naturalna była dla mnie praca w trybie przetwarzania tekstów. Dla osoby zaznajomionej uprzednio z jakimś procesorem tekstowym, np. w WordStarem, operowanie w tym trybie jest dość łatwe. Wykorzystanie możliwości formatowania i drukowania tekstu nie sprawia specjalnych trudności. Niestety, nie udało mi się, być może wskutek niewiedzy, odczytać w WordStarze pliku tekstowego zapisanego przez Computex — i odwrotnie. W każdym razie, nie mogłam samodzielnie znaleźć żadnej informacji, jak to zrobić.

Dość łatwo szło mi zaznajamianie się z podstawowymi zasadami pracy w trybie **SHEET**, a więc — ze sposobem wykorzystania arkuszy kalkulacyjnych. Kiedy jednak doszło do próby ich użycia jako bazy danych w trybie **FORM**, napotkałam trudności i zarzuciłam te próby. Bardzo zachęcająco brzmiąca deklaracja, że pakiet Computex zawiera program tłumaczący w obie strony zadania systemu dBASE, także pozostała nie sprawdzona.

Próbując użyć pakietu w trybie **GRAPH** przeżyłam niemiłe rozczarowanie, gdy okazało się, że nie może on sporządzać rysunków, lecz tylko wykresy. Nie jest to więc pełna grafika, lecz tylko tzw. business graphics, tj. w zasadzie wizualizacja danych. Przed właściwym wykorzystaniem tych danych trzeba opanować posługiwanie się arkuszami, ponieważ na ich podstawie tworzy się wykresy. Choć to zadanie nie przekraczało moich możliwości, napotkałam inne trudności. Okazało się, że dla tego trybu pracy szczególnie istotny jest rodzaj posiadanego sprzętu. Ponieważ nie miałam drukarki firmy Hewlett-Packard, a sama nie mogłam sobie poradzić z zainstalowaniem pakietu do własnej drukarki, niewiele mogłam zobaczyć.

Praca w trybie **COMM**, tzn. komunikacja z innym komputerem, nie była możliwa, o czym lojalnie uprzedzono w instrukcji będącej wprowadzeniem do użytkowania Computexa. Stwierdziłam jednak, że samo przygotowanie komunikacji od strony programowej jest dość łatwe dla początkującego użytkownika. Dziwne wydało mi się tylko to, że przy wymuszeniu transmisji, lecz braku faktycznego połączenia, komunikat:

Czekam na połączenie...

trwał nieskończenie długo, bez sygnału przeterminowania, tzn. time-out'u.

Spośród bardziej złożonych funkcji pakietu nie sprawdziłam, jakie możliwości zapewnia posługiwanie się oknami. Uważam, że rozsądne byłoby korzystanie z okien dopiero po opanowaniu każdego z trybów pracy oddzielnie. Nie używałam też języka poleceń, choć stosowanie makropoleceń jest z pewnością konieczne, jeżeli pragnie się wykorzystać całą siłę Computexa. Z powodu zbytniej złożoności problemu nie próbowałam też sprawdzić, jak można dołączyć własny program aplikacyjny.

Ważnym składnikiem oprogramowania, wręcz decydującym o łatwości posługiwania się każdym programem, jest jego dokumentacja. Ocenianie dokumentacji Computexa

tylko na podstawie 33-stronicowego wprowadzenia w języku polskim byłoby z pewnością nieuczciwe. Jest jednak oczywiste, że samo wprowadzenie, nawet uzupełnione samouczkiem elektronicznym (HELP), nie wystarcza do sprawnego używania pakietu. Świadome korzystanie z tego oprogramowania wymaga posiadania pełnego opisu. Jednakże podręczniki do Symphony (Introduction, How-To Manual, Reference Manual i Quick Reference), niepełny podręcznik polski i samouczek stanowią zbyt rozwlekły materiał, szczególnie jeśli w jednym tomie występują odsyłacze do drugiego, który z kolei zawiera odsyłacze powrotne.

Jako filolog nie mogę też nie zwrócić uwagi na sposób napisania dokumentacji polskiej, choć informatycy wieloma sprawami mogą się nie przejmować. Sformułowania „użyj program", „użyj parametry", „operowanie na danych" i „operowanie na plikach" (nie mówiąc o takich błędach, jak „adjustacja" lub „przesył" — powinno być „adiustacja" i „przesyłanie") świadczą o niedostatecznej pracy nad tekstem polskojęzycznym (czasownik „używać" rządzi dopełniaczem, a czasownik „operować" — narzędnikiem lub biernikiem). Nie jestem też przekonana, czy warto spolszczać tekst nie używając polskiego liternictwa.

* * *

Biorąc pod uwagę fakt, że testując Computex wykonywałam coś w rodzaju programistycznej „nauki chodzenia", nie można powiedzieć, iż przebadałam go dogłębnie. Doświadczony programista miałby o nim wiele więcej do powiedzenia, szczególnie jeśli używał przedtem pakietu 1-2-3 firmy Lotus, którego rozwiniętą odmianę (wzbogaconą o przetwarzanie tekstów i komunikację) stanowi Symphony.

Uważam jednak, że przy dobrze określonym kryterium oceny przebadanie pakietu Computex nawet w tak szczątkowy sposób jest korzystne. Można bowiem zakomunikować dostawcom i użytkownikom, jakie pułapki czyhają na nich przy adaptowaniu lub spolszczaniu oprogramowania pochodzącego z zagranicy.

Computex w żadnym razie nie jest czymś, co „umożliwia pracę nawet takiemu użytkownikowi, który po raz pierwszy zetknął się z tym pakietem". Jest bardzo dobrym narzędziem, ale dla zaawansowanych programistów. Przyczynę takiej oceny wyjaśnił, jak sądzę najlepiej, jeden z recenzentów czasopisma PC World, który napisał: „Oczywiście Symphony ma niezliczoną liczbę różnych możliwości. ... Problem polega jednak na uzyskaniu do nich dostępu... Symphony zmusza użytkownika do wykonania dużej liczby prac przygotowawczych przed uzyskaniem jakichkolwiek wyników" (PC World, p. 138, July 1984).

M. M. ZALEWSKA

WOJCIECH MOKRZYCKI
Instytut Maszyn Matematycznych
Warszawa

System programowania grafiki komputerów PSG 2.2

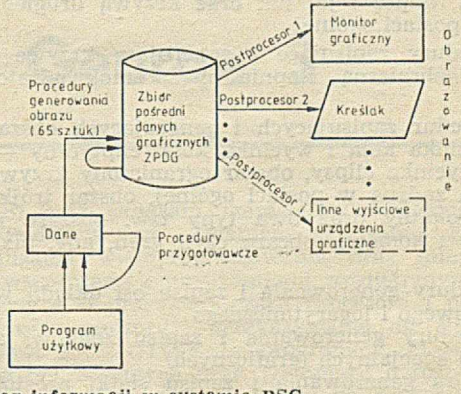
W końcu lat siedemdziesiątych, w Pracowni Grafiki Komputerowej IMM opracowano sprzętowo niezależny system programowania grafiki komputerów pod nazwą Procedury Syntezy Grafiki (PSG) [1].

Pierwszą jego implementację wykonano na minikomputerze MERA 400 z systemem operacyjnym SOM-3. W kolejnych latach system udoskonalono i rozbudowano. Sprzedano go do kilkunastu ośrodków komputerowych na terenie całego kraju. Został zaimplementowany również na minikomputerze SM-4 z systemem operacyjnym DOS RW. Obecnie jest rozprzestrzeniona jego najdoskonalsza, trzecia z kolei SM-owska generacja, oznaczona PSG 2.2.

Myślą przewodnią koncepcji systemu PSG było założenie, że powinien to być system do programowania obrazów (rysunków), a nie urządzenia graficznego. Występujące z konieczności programy, zależne od sprzętu, należało zredukować do minimum i wydzielić w postaci niezależnej warstwy programowej, zwanej warstwą postprocesorów (logicznych sprzęgów jądra systemu z urządzeniami graficznymi), które dostosowują wewnątrzsystemową strukturę obrazu do sprzętowych funkcji określonego urządzenia graficznego.

Powstał więc ujednolicony system programowania grafiki komputerów, którego mechanizmy są niezależne od sprzętu, systemu operacyjnego i języków programowania. Może być on zaimplementowany w dowolnym uniwersalnym języku programowania, na dowolnym uniwersalnym komputerze, do którego jest dołączone dowolne wyjściowe urządzenie graficzne.

Podstawowymi elementami systemu PSG są trzy grupy procedur, tzn. procedury przygotowawcze, generowania obrazu, postprocesorów, oraz zbiór pośredni danych graficznych ZPDG (rys. 1).



Rys. 1. Obieg informacji w systemie PSG

Procedury przygotowawcze

W skład tej grupy wchodzą procedury: definiujące (19 szt.) oraz transformacyjne (18 szt.).

Procedury definiujące pozwalają na podstawie różnych danych, wziętych z programu użytkowego bądź wprowadzonych z zewnątrz, obliczyć inne dane i sprowadzić je do postaci standardowej, opisującej pewne pierwotne elementy graficzne — Pierwotniki Graficzne (PG), z których można składać obraz.

Wśród procedur definiujących wyróżnia się 6 grup, w każdej z nich występują procedury odnoszące się do pierwotników tego samego typu:

- procedur definiujących punkt,
- 5 procedur definiujących odcinek prostej,
- 4 procedury definiujące okrąg,
- 2 procedury definiujące łuk okręgu,
- procedura definiująca krzywą drugiego stopnia, przechodzącą przez pięć punktów,
- procedura wyznaczania azymutu prostej.

Procedury transformacyjne umożliwiają wykonywanie podstawowych przekształceń na pierwotnikach graficznych PG: obrotu o dowolny kąt względem dowolnego punktu płaszczyzny, skalowania przez dowolny dodatni współczynnik oraz przesunięcia o dowolny wektor.

Parametry transformacji mogą występować bezpośrednio w procedurze jako parametry formalne bądź też mogą być przechowywane w tablicy, która sama jest parametrem formalnym procedury.

Procedury transformacyjne można podzielić na cztery grupy:

- 10 procedur transformacji punktu i wektora oraz ciągu punktów i wektorów,
- procedury transformacji okręgu oraz jego łuku,
- 2 procedury transformacji segmentu krzywej drugiego stopnia,
- 2 procedury transformacji segmentu krzywej stopnia n (n>2).

Procedury generowania obrazu

W skład tej grupy wchodzą procedury generujące (50 szt.) oraz obsługi zbioru pośredniego ZPDG (15 szt.).

Procedury generujące umożliwiają składanie pierwotników graficznych PG w większe jednostki — obiekty — zapamiętywane w specjalnym zbiorze, zwanym Zbiorem Pośrednim Danych Graficznych (ZPDG). Pierwotniki PG zostały tak wybrane, aby łatwo było składać z nich obiekty, od najprostszych po najbardziej złożone.

Wśród procedur generujących można wyróżnić następujące grupy:

- procedury zapisujące (w ZPDG) znaczniki (punkty) bądź ich ciągi określone w sposób bezwzględny i względny
- 6 procedur zapisujących odcinki, ciągi odcinków (tzn. łamaną) oraz zbiory wektorów — określonych bezwzględnie i względnie
- 5 procedur zapisujących krzywe drugiego stopnia: okrąg i jego łuk, elipsę i jej łuk oraz krzywą drugiego stopnia zadaną w postaci ogólnej
- 3 procedury zapisujące i generujące krzywe wyższych stopni: algebraiczną stopnia n, średniokwadratową oraz gładką
- 11 procedur zapisujących i generujących obszary: kołowy — odcinka koła i wycinka koła, eliptyczny — odcinka elipsy i wycinka elipsy, obszar ograniczony krzywą drugiego stopnia zadaną w postaci ogólnej, obszar trójkątny, obszar prostokątny oraz dwa typy tzw. obszaru grelowego (złożonego z komórek obrazu czyli greli, ang. pixel-picture elements)
- 2 procedury generowania i zapisu osi układu współrzędnych: liniowego i logarytmicznego
- — 2 procedury generowania i zapisu znaków: alfanumerycznych i specjalnych (graficznych)
- — procedura generowania i zapisu siatki prostokątnej
- — 2 procedury generowania i zapisu wykresu liniowego bądź słupkowego
- — 1 procedura zadająca numer bloku informacyjnego w ZPDG
- — 13 procedur określających cechy (atrybuty) obrazowania.

Z każdym pierwotnikiem graficznym może być związana pewna cecha: jego typ, kolor, rodzaj linii, wzór wypełnienia wnętrza.

Procedury obsługi ZPDG umożliwiają kreowanie, usuwanie oraz dokonywanie różnych przekształceń na obiektach w nim zapisanych. Procedury te można podzielić na następujące grupy, realizujące zbliżone do siebie przekształcenia:

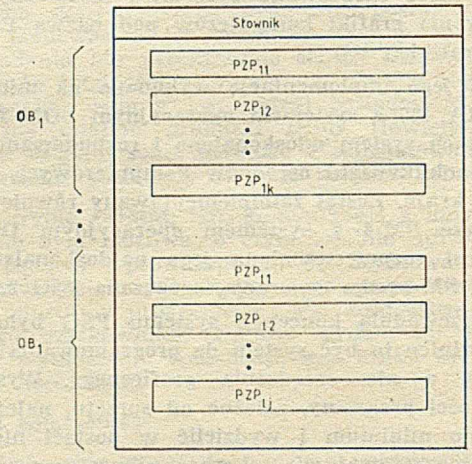
- procedura inicjująca ZPDG
- 3 procedury generowania (zapisu do ZPDG) nowych bądź rozszerzenie istniejących obiektów
- 5 procedur zmiany obiektu: usuwania obiektu, scalania dwóch obiektów w jeden, zmieniające nazwę obiektu bądź jego układ współrzędnych oraz przenoszące obiekt do innego zbioru
- 4 procedury transformacji obiektu: przesunięcia o dowolny wektor, skalowania przez dowolny dodatni współczynnik, obrotu względem dowolnego punktu płaszczyzny oraz odbicia lustrzanego względem prostej
- 2 procedury informacyjne: wypisująca zawartość ZPDG oraz podająca ogólne informacje o tym zbiorze.

Zbiór Pośredni Danych Graficznych

ZPDG jest elementem systemu PSG warunkującym — w zasadzie — jego niezależność sprzętową. Stanowi wewnątrzsystemowe archiwum obiektów (obrazu). Przechowywane w nim obiekty graficzne są zbudowane z jednostek logicznych, zwanych Pierwotnikami Zbioru Pośredniego (PZP). Postacie (typy) PZP są zasadniczo odmienne od postaci pierwotników graficznych PG, z których obiekty w ZPDG są składane. Zostały one wybrane według innych niż PG przesłanek: są najbardziej złożonymi elementami graficznymi, które mogą być sprzętowo realizowane w dostępnych urządzeniach graficznych. Pierwotnikami zbioru pośredniego PZP są następujące elementy geometryczne:

— znaczniki (punkty)
— odcinki prostych (łamana)
— okręgi (łuki okręgów)
— znaki alfanumeryczne
— znaki specjalne
— cechy (atrybuty) danych graficznych
— wykres (liniowy bądź słupkowy)
— pola grelowe (komórkowe)
— pola prostokątne.

ZPDG jest zbiorem obiektów OB identyfikowanych przez nazwę, którą nadaje użytkownik, i przez numer jako kolejny element zbioru (rys. 2). Obiekt tworzy się za pomocą procedur generujących i procedur obsługi zbioru. W ZPDG jest on niepodzielnym elementem graficznym (wszystkie operacje w ZPDG są realizowane na tych właśnie elementach). Każdy z obiektów ma swój układ współrzędnych — współrzędne obiektu.



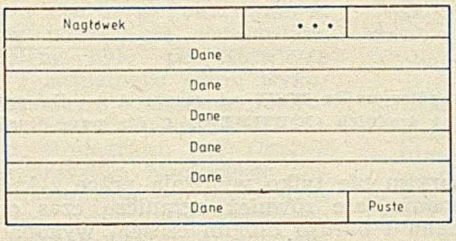
Rys. 2. Ogólna budowa ZPDG (OB — obiekty, PZP — pierwotniki zbioru pośredniego)

ZPDG rozważany jako element systemu jest zbudowany ze słownika oraz kolejno zapisanych pierwotników zbioru pośredniego PZP. Słownik zawiera informacje o miejscu zapisu kolejnych obiektów w zbiorze. Pierwotnik PZP jest ciągiem danych jednego typu, poprzedzonych nagłówkiem (rys. 3). Rozważany jako element systemu stanowi ciąg rekordów. W pierwszym rekordzie jest nagłówek, podzielony na pola zawierające informacje o danych tworzących PZP. Pierwsze dwa pola S_1 i S_2 w nagłówkach wszystkich typów PZP określają:

S_1 — kod typu pierwotnika PZP
S_2 — nazwę procedury generującej dany typ pierwotnika PZP.

Kolejne cztery pola — S_3, S_4, S_5 i S_6 — nagłówka zawierają dla większości typów PZP informację na temat tych samych wielkości:

S_3 — rozmiar przestrzeni bądź liczba znaków do wpisania
S_4 — postać danych dla różnych typów PZP
S_5 — liczba rekordów zawierających dane graficzne dla określonego typu PZP
S_6 — liczba słów zawartych w ostatnim rekordzie danych.



Rys. 3. Ogólna budowa pierwotnika PZP

Dla niektórych typów PZP są wypełnione również dalsze pola nagłówka. Dla PZP zawierającego cechy (atrybuty) obrazowania (kod typu=6) nagłówek składa się tylko z pól S_1 i S_2.

Postprocesory

Są to programy analizujące wewnątrzsystemowe archiwum niezależnie od sprzętu zdefiniowanych obiektów (złożonych z pierwotników PZP) i ich przetwarzanie na postać rozkazów i kodów interpretowanych przez poszczególne typu urządzeń graficznych, na których obiekty te powinny być obrazowane. W systemie przewidziano dwa postprocesory dla każdego typu urządzenia obrazującego. Jeden z postprocesorów wywołuje obiekty przez ich nazwę (postprocesor jednoparametrowy), drugi — obiekty oznaczone w ZPDG numerami od v do q (postprocesor dwuparametrowy). Implementacja systemu PSG na minikomputerze MERA 400 zawiera postprocesory dla: monitora MERA 7954 oraz MERA 7954V, kreślaka KL-2, kreślaka Benson 122 (w połączeniu bezpośrednim oraz za pośrednictwem taśmy magnetycznej z MERĄ 400), kreślaka DIGIGRAF 1612 (podłączonego bezpośrednio oraz za pośrednictwem taśmy papierowej), kreślaka NE 2000. Implementacja PSG na minikomputerze SM4 zawiera postprocesor jedynie dla monitora MERA 7954V. Planuje się dołączenie do systemu postprocesorów na powstające nowe urządzenia graficzne.

Kolejna generacja systemu PSG będzie miała również możliwości wprowadzania rysunków do komputera, za pośrednictwem półautomatycznego dyskretyzatora rysunków i ich przekształcania do postaci wymaganej przez zbiór ZPDG, a także wprowadzania obrazów za pomocą dyskretyzatora rastrowego (z zastosowaniem fotokamery) i wstępnego przetwarzania obrazów (dyskretnych) oraz obrazowania na monitorach.

LITERATURA,

[1] Bronowska M., Mokrzycki W.: Oprogramowanie rastrowego monitora graficznego w systemie MERA 400. Archiwum opracowań IMM, str. 91, Warszawa, grudzień 1971.

PRZEMYSŁAW ROKITA
Instytut Informatyki
Politechnika Warszawska

Nowa wersja programu szeregującego wysokiego poziomu dla systemu operacyjnego George 3

Najważniejszym celem szeregowania prac w systemie cyfrowym jest minimalizacja kosztów pracy komputera oraz maksymalne wykorzystanie czasu oczekiwania użytkowników. Najczęściej zakłada się, że jednostkowy koszt czasu oczekiwania na reakcję komputera jest identyczny dla wszystkich użytkowników. Przy takim założeniu problem polega na zredukowaniu sumy czasów oczekiwania wszystkich użytkowników. Każda praca na czas swojego wykonania opóźnia prace ustawione za nią w kolejce. Jest więc bardzo ważne, aby liczba oczekujących prac była jak najmniejsza. Dlatego większość stosowanych algorytmów szeregowania nadaje wysoki priorytet pracom o krótkim czasie wykonania — kosztem prac o długim czasie wykonania.

W systemie komputerowym znajduje się wiele zasobów podlegających podziałowi. Do najważniejszych należą: czas procesora, pamięć operacyjna i pamięć masowa. Każdy z tych zasobów wymaga odpowiedniego systemu podziału między użytkowników. W systemie GEORGE-3 podział ten jest realizowany przez następujące programy: program EXECUTIVE, program szeregujący niskiego poziomu (LLS) oraz program szeregujący wysokiego poziomu (HLS). Podstawowym zadaniem programu EXECUTIVE jest realizacja wieloprogramowania oraz inicjowanie fizycznej współpracy z urządzeniami zewnętrznymi. Program LLS decyduje o przydziale czasu procesora i pamięci operacyjnej oraz nadzoruje przekazywanie do wykonania — programowi EXECUTIVE — programów, które należą do aktualnie realizowanych prac. Natomiast o rozpoczęciu wykonywania pracy i przekazaniu jej do programu LLS decyduje program HLS.

W większości instalacji pracujących pod kontrolą systemu operacyjnego GEORGE-3, standardowy program szeregujący wysokiego poziomu (HLS) dostarczany przez producenta, nie jest wykorzystywany. Dzieje się tak częściowo dlatego, że przy generacji systemu makropolecenie HLSOFF jest przyjmowane przez domniemanie, a przede wszystkim dlatego, że standardowy program HLS nie ma żadnego wpływu na poprawę przepustowości systemu. Realizuje on strategię szeregowania FIFO i gdy system pracuje ze standardowym programem HLS (jak również bez niego), kolejność uruchamianych prac jest taka sama. Program ten

zajmuje wraz z budowanymi przez siebie kolejkami ok. 4 K słów pamięci operacyjnej, wymaga budowania przez system kolejki prac oczekujących na obsługę przez program szeregujący, a także sam potrzebuje czasu procesora. Tak więc używanie standardowego programu HLS nie jest ekonomicznie uzasadnione, natomiast w systemie została przewidziana możliwość napisania przez użytkownika nowego programu HLS, dostosowanego do potrzeb danej instalacji.

NOWA WERSJA PROGRAMU HLS

Opracowana nowa wersja programu HLS (program RHLS) ma następujące zalety w stosunku do wersji standardowej:

- odzyskuje okres oczekiwania prac w pełni uruchomionych na:
  - odzyskanie pliku (ang. retrieve),
  - otwarcie dostępu do pliku (zwolnienie pliku przez inną pracę),
  - dołączenie rzeczywistego urządzenia zewnętrznego,
  - załadowanie taśmy magnetycznej.

  Oczekiwanie takie trwa z reguły kilka minut i w tym czasie krótsza praca może zostać całkowicie wykonana;

- realizuje przeplatanie prac obciążających głównie urządzenia zewnętrzne z pracami obciążającymi CPU. Jeżeli praca w pełni uruchomiona w danym momencie ma współczynnik WUZ większy od 4 (współczynnik WUZ obliczany jest przez program RHLS i jest równy iloczynowi liczby typów urządzeń zewnętrznych żądanych przez zadanie z sumaryczną liczbą urządzeń zewnętrznych żądanych przez zadanie), to bezpośrednio po niej jest w pełni uruchamiana praca o wskaźniku WUZ $<=2$, dla której wartość stosunku JT do WWT (czas przewidziany na pracę dzielony przez ważony czas oczekiwania pracy w kolejce RHLSQ) jest najmniejsza;

- aby ograniczyć czasochłonne „wymiatanie" dużych programów z pamięci operacyjnej, program RHLS uzależnia wielkość współczynnika CPI (na podstawie tego współczynnika wyznaczana jest wielkość kwantu czasu procesora przyznawana zadaniu) od wielkości zadania, według następującego wzoru:

$$CPI = 2 * SIZE * (26 - UR)$$

gdzie: SIZE jest wymaganą przez pracę wielkością pamięci operacyjnej w słowach, a UR — pilnością pracy ($A = 0$, $B = 1$, ..., $Z = 25$). Programy o większej wartości współczynnika CPI otrzymują od systemu większy kwant czasu, dzięki czemu w czasie wykonania są one rzadziej przenoszone z pamięci operacyjnej do masowej i z powrotem;

- stosuje oryginalny, bardziej wydajny algorytm szeregowania (program standardowy realizuje strategię FIFO).

ZAŁOŻENIA ALGORYTMU RHLS

Kolejka prac oczekujących na obsługę uporządkowana jest według wartości współczynnika ważonego czasu oczekiwania WWT, który jest obliczany w następujący sposób:

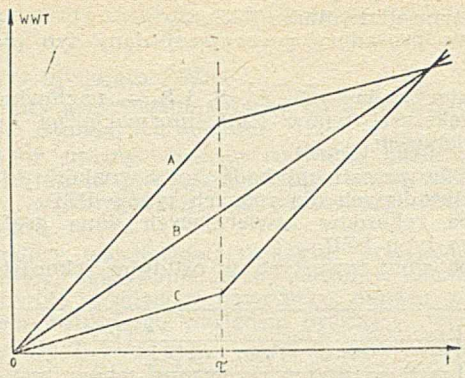
$$WWT = \left(\frac{\tau}{JT} * f_1(\Delta t) + \frac{JT}{\tau} * f_2(\Delta t)\right) * (26 - UR)$$

gdzie: JT oznacza czas przewidziany na pracę, UR — pilność pracy ($A = 0$, $B = 1$, ..., $Z = 25$), Δt — czas przebywania pracy w systemie, $\tau = LIM * JOBTIME$, LIM — constans, JOBTIME — parametr instalacyjny określający standardowy czas przetwarzania pracy, a ponadto

$$f_1(\Delta t) = \begin{cases} \Delta t & \text{dla} \quad \Delta t < \tau \\ 0 & \text{dla} \quad \Delta t >= \tau \end{cases}$$

$$f_2(\Delta t) = \begin{cases} 0 & \text{dla} \quad \Delta t < \tau \\ (\Delta t - \tau) + \frac{\tau}{JT} * \tau & \text{dla} \quad \Delta t >= \tau \end{cases}$$

Zależność WWT(t) ma postać przedstawioną na rysunku.



Zależność współczynnika WWT od czasu: A — dla prac krótkich, B — dla prac średnich (JOBTIME=), C dla prac długich

Taki algorytm nie tylko wyróżnia prace z krótkim czasem wykonania, ale również ogranicza czas oczekiwania prac z długim i bardzo długim czasem wykonania. Jeżeli długa praca pozostaje w systemie dostatecznie długo, to w końcu osiągnie tak wysoki priorytet, że będzie obsłużona przed innymi krótszymi pracami. Tak więc szeregowanie prac zgodne z proponowanym algorytmem wydajnie ogranicza czas oczekiwania większych prac, przy zachowaniu zalet wynikających z preferowania prac krótkich. Poza tym, szeregowanie za pomocą tego algorytmu jest bardziej „sprawiedliwe" dla wszystkich prac, niż na przykład w wypadku algorytmu FIFO — tzn. występują mniejsze różnice ważonego czasu oczekiwania między pracami o różnych długościach.

* * *

Program RHLS od początku 1985 r. przeszedł szereg badań i testów, a następnie został wdrożony do eksploatacji na komputerze ODRA 1305 w Ośrodku Obliczeniowym Instytutu Informatyki Politechniki Warszawskiej. W sposób ciągły jest eksploatowany od maja 1985 r. przy dużym obciążeniu systemu wynoszącym ok. 300 zadań na jedną zmianę.

W tym czasie można było wyraźnie zaobserwować zalety zrealizowanego programu szeregującego, którego użytkowanie w zauważalny sposób poprawiło przepustowość systemu zarówno dla prac wsadowych jak i konwersacyjnych, zmniejszając czas reakcji systemu w porównaniu z pracą bez RHLS. Ciągła eksploatacja programu pozwoliła go dokładnie przetestować i stwierdzić, że nie zawiera on błędów.

Przewidziano stałą eksploatację tego programu i przeprowadzenie badań, które umożliwiłyby wyciągnięcie wniosków dotyczących zmian strategii działania RHLS w zależności od charakterystyki strumienia zadań. Pozwoli to także na wykonanie dokładniejszej analizy ilościowej i porównawczej jego wpływu na poprawę przepustowości systemu.

Stała sprzedaż egzemplarzy

MIKROKLANU

prowadzona jest

w Klubie Prasy Technicznej

w Warszawie

ul. Mazowiecka 12, tel. 27-43-65

Systemy mikrokomputerowe w zarządzaniu gospodarką

W styczniu 1986 r. w Pokrzywnej k. Głuchołazów odbyło się seminarium, którego celem było przedstawienie możliwości i aktualnego stanu wykorzystania systemów mikrokomputerowych w zarządzaniu gospodarką. Organizatorem seminarium był Opolski Oddział Towarzystwa Naukowego Organizacji i Kierownictwa oraz Spółdzielnia Pracy Usług Informatycznych i Organizacyjnych INFOPOL z Opola.

Całość przedstawionych referatów i komunikatów można podzielić na dwie grupy. Pierwsza dotyczyła konkretnych zastosowań systemów mikrokomputerowych w automatyzacji prac biurowych lub w pracach konstrukcyjno-technologicznych, druga zaś — zagadnień związanych z oprogramowaniem narzędziowym mikrokomputerów (system operacyjny PC-DOS, języki C i Modula-2, dBASEIII, sieci lokalne).

Seminarium rozpoczęto referatem nie mieszczącym się w żadnej z wymienionych grup, charakteryzującym polski rynek mikrokomputerów profesjonalnych (W. Czajkowski, Opole). Żądane charakterystyki typowej konfiguracji mikrokomputera profesjonalnego najlepiej spełnia sprzęt oferowany przez firmy polonijno-zagraniczne, a zwłaszcza mikrokomputery 16-bitowe kompatybilne z IBM PC/XT, jak EMIX86P, CS88PC i MP-86. Jeśli chodzi o wytwórców państwowych, to oferowany przez nich sprzęt, ze względu na bariery technologiczne, ogranicza się w zasadzie do mikrokomputerów 8-bitowych (PSPD-90, MK-45, ELWRO-523, ComPAN8 oraz Robotron z NRD) z ubogim oprogramowaniem. Choć znaczną zaletą jest dobrze zorganizowany serwis techniczny dla tego sprzętu, to dużym minusem jest bardzo długi okres oczekiwania na dostawę (rok lub dwa lata).

W pierwszej grupie tematycznej wyróżniał się referat omawiający **wykorzystanie mikrokomputerów do automatyzacji prac biurowych** (W. Czajkowski). Przedstawiono w nim model systemu informatycznego mającego usprawnić działalność przedsiębiorstwa. Istotnym zagadnieniem wyeksponowanym w referacie jest wcześniejsze określenie zadań, jakie mają być realizowane i dopasowanie do nich możliwości kupowanego sprzętu komputerowego. W referacie zawarto również krótką charakterystykę dostępnego oprogramowania narzędziowego wykorzystywanego w pracach biurowych, takiego jak programy zarządzania bazami danych, przetwarzania tekstów, kalkulacyjne, graficzne i teletransmisyjne.

Bardzo wymierne korzyści płynące z **zastosowania mikrokomputerów** przedstawiono w referacie dotyczącym ich wykorzystania w **pracach konstrukcyjno-technologicznych** (L. Wolański). Biorąc pod uwagę tylko podstawowe zalety wynikające z wykorzystania mikrokomputera, takie jak:

— skrócenie cyklu projektowania,
— poprawa jakości i stylu pracy,
— uatrakcyjnienie pracy w biurze,
— utworzenie archiwum komputerowego,

trudno sobie wyobrazić przyszłość biur konstrukcyjnych pracujących bez systemów mikrokomputerowych.

Do tej grupy tematycznej należy również zaliczyć dwa komunikaty **„Mikrokomputerowy system ewidencji finansowo-księgowej"** i **„Wielodziedzinowy mikrokomputerowy system informatyczny dla przedsiębiorstw i instytucji"**. Pierwszy system został zrealizowany przez Zakład Elekroniki, Informatyki i Organizacji ELINOR — Spółdzielnia Pracy w Gliwicach. Program ten, działający pod nadzorem systemu operacyjnego CP/M2.2, realizuje następujące funkcje:

- prowadzenie kartoteki kont,
- rejestracja transakcji finansowych,
- aktualizacja stanów kont na bieżąco.

Drugi system opracowany przez Spółdzielnię INFOPOL, ma na celu wspomaganie zarządzania w przedsiębiorstwach przy użyciu mikrokomputerów osobistych. System ten zawiera następujące podsystemy dziedzinowe:

- techniczne przygotowanie produkcji,
- planowanie i kontrola produkcji,
- gospodarka materiałowa,
- gospodarka wyrobami gotowymi,
- gospodarka środkami trwałymi,
- kadry,
- zatrudnienie i płace,
- ewidencja finansowo-księgowa,
- koszty własne produkcji,
- wewnątrzzakładowy rozrachunek gospodarczy,
- gospodarka transportem samochodowym,
- obsługa zakładowych komórek organizacji politycznych, związkowych i społecznych,
- podsystem informowania kierownictwa.

Pozostała część referatów i komunikatów dotyczyła oprogramowania narzędziowego. Referat **„System operacyjny PC-DOS"** (J. Zalewski, Warszawa) zapoznał uczestników z podstawowymi pojęciami systemu operacyjnego, pod którego nadzorem pracują komputery IBM PC i ich liczne odmiany. Konieczność zapoznania się z tym systemem wśród użytkowników sprzętu mikrokomputerowego wynika nie tylko z faktu przyjęcia w krajach RWPG mikrokomputera IBM PC jako standardu, ale również z powodu ewolucji tego systemu w kierunku systemu Unix. Poza programem uczestników seminarium zapoznano również z językiem C — zyskującym coraz większą popularność.

Zagadnieniom poruszonym przy omawianiu automatyzacji prac biurowych, poświęcono referat nt. **Lokalnych sieci komputerowych** (J. Zalewski). Przytaczając słowa autora można stwierdzić, że „sens stosowania lokalnych sieci komputerowych w zarządzaniu polega na stworzeniu lokalnego dostępu do mocy obliczeniowej wszystkim wymagającym tego jednostkom wewnątrz przedsiębiorstwa, bez konieczności używania dużego komputera". W praktyce chodzi tu głównie o dostęp do urządzeń, których ze względu na wysoki koszt nie można zainstalować przy każdym mikrokomputerze, takich jak bardzo pojemne pamięci dyskowe i urządzenia wyprowadzania danych (monitory graficzne, pisaki, drukarki graficzne).

W chwili obecnej najbardziej popularnymi rozwiązaniami sieci lokalnych są omówione już w »Informatyce«:

- sieć Ethernet firmy Xerox, opracowana przy współudziale firm Intel i DEC, mająca zastosowanie głównie w pracach biurowych,
- sieć Proway, opracowana przez IEC do stosowania w przemyśle. W Polsce prace nad realizacją sieci lokalnych prowadzi się głównie w Przemysłowym Instytucie Automatyki i Pomiarów w Warszawie (sieci Proway) oraz Politechnice Śląskiej (sieci Ethernet).

Możliwościom wytwarzania oprogramowania użytkowego w języku Modula-2 i stworzenia technologii produkcji oprogramowania na zasadach pracy zespołowej poświęcony był komunikat **„Wytwarzanie oprogramowania użytkowego mikrokomputerów w języku Modula-2"** (E. Westerowska, T. Michałowski, Warszawa).

Natomiast referat **„Systemy zarządzania bazami danych w systemach mikrokomputerowych"** (K. Kubiak, EMIX) zapoznał uczestników z podstawowymi wymaganiami stawianymi organizacji baz danych, podstawowymi typami struktur danych stosowanych do reprezentacji baz danych oraz z systemem zarządzania relacyjną bazą danych dBASEIII, będącym obecnie standardem światowym dla mikrokomputerów 16-bitowych.

W referacie omawiającym relację między operatorem mikrokomputera a samym mikrokomputerem, pt. **„Dialog człowieka z maszyna w systemach mikrokomputerowych"** (B. Koska, Opole), zwrócono uwagę na konieczność projektowania systemów dialogowych (wymiany informacji między komputerem i operatorem) tak, aby mógł z nich swobodnie korzystać użytkownik tylko przyuczony do obsługi sprzętu.

Kilkudziesięciu uczestników seminarium z przedsiębiorstw Polski południowej wyjechało z Pokrzywnej, jak należy sądzić, z podstawowym zasobem informacji niezbędnych do właściwego stosowania komputerów w zarządzaniu. W przyszłości zapewne zwiększy się zarówno częstotliwość, jak i zasięg podobnych konferencji, choćby ze względu na konieczność poszerzania wiedzy informatycznej przez kadry kierownicze przedsiębiorstw.

MACIEJ ADAMCZYK

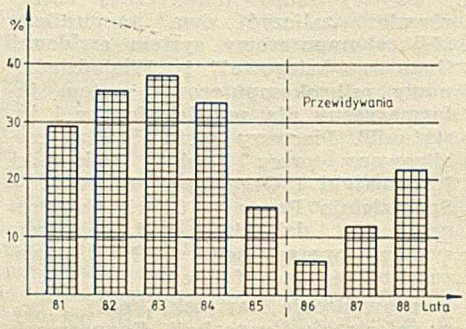
IBM — numer 1?

Na liście 1000 największych pod względem wartości rynkowej przedsiębiorstw w 1985 r. w USA, zamieszczonej w specjalnym wydaniu BUSINESS WEEK, pierwsze miejsce zdecydowanie zajmuje International Business Machines Corp. — znane jako IBM. Kolejna firma z przemysłu komputerowego — Hewlett-Packard Co. (HP) — zajmuje 25 pozycję, a Digital Equipment Corp. (DEC) pozycję 40. Na liście znalazło się wiele innych znanych firm produkujących komputery, układy logiczne oraz prowadzących usługi w zakresie oprogramowania serwisu (tab. 1).

SYTUACJA NA RYNKU KOMPUTEROWYM

Rok 1985 przyniósł jednak dalszy spadek przyrostu dochodów w przemyśle sprzętu komputerowego. Po bardzo dobrym 1983 r., kiedy wzrost dochodów wynosił ponad 37% w stosunku do roku poprzedniego, 1985 r. przyniósł 15% wzrostu, a przewidywania na 1986 r. wynoszą tylko 7,4% wzrostu. Dopiero 1987 r. powinien być nieco lepszy (rys. 1).

Na ten stan złożyło się wiele przyczyn. Ostre wahania i spadek kursu dolara, reforma systemu podatkowego i spadek wydatków na inwestycje o 1% w 1985 r., w całym przemyśle zmniejszyło zapotrzebowanie na nowy sprzęt. Również ogólne zestarzenie się dotąd oferowanego sprzętu, obniżka kosztów produkcji oraz ostra konkurencja, szczególnie na rynku mikrokomputerowym, zmusiły firmy do znacznych obniżek cen.

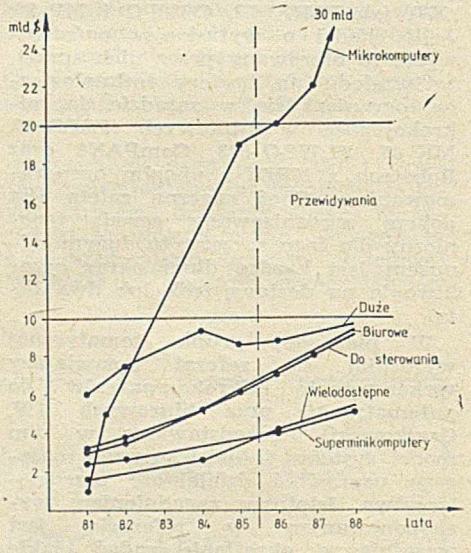


Rys. 1. Roczne przyrosty dochodów ze sprzedaży komputerów

W sumie przewiduje się, że sprzedaż w 1986 r. zamknie się kwotą 47,5 mld dol. przy ogólnoświatowych dochodach z oprogramowania i serwisu osiągających sumę 300 mld dol.

Dotychczasowy gwałtowny rozwój informatyki nie mógł w tej samej skali trwać wiecznie. Użytkownicy dużych systemów, super- i minikomputerów oraz milionów mikrokomputerów stanęli przed problemem ogarnięcia tego całego zbioru różnorodnych architektur i koncepcji oprogramowania. Podstawowym dążeniem organizacji użytkujących komputery jest obecnie posiadanie sprawnych sieci komputerowych wchłaniających całość sprzętu. Pomijając problemy techniczne w dopasowaniu różnego sprzętu i oprogramowania do jednolitej struktury sieci, problemem jest pokonanie barier psychologicznych u dotychczasowych użytkowników pojedynczych procesorów biurowych oraz mikrokomputerów osobistych w korzystaniu z takich sieci. Do korzystania z superkomputerów i superminikomputerów przez sieć ze swoich mikrokomputerów osobistych nieco lepiej przygotowani są naukowcy i inżynierowie. Duży i chłonny rynek tworzą zastosowania w sferze automatyzacji produkcji.

W rezultacie na 1986 r. przewiduje się dalszą stagnację sprzedaży dużych maszyn i mikrokomputerów, a jedynie niewielki przyrost sprzedaży superminikomputerów, komputerów biurowych, wielodostępnych mikrokomputerów oraz systemów sterowania (rys. 2).



Rys. 2. Dochody ze sprzedaży różnych typów komputerów

Tabela 1. Wybrane przedsiębiorstwa przemysłu komputerowego z listy 1000 największych przedsiębiorstw w USA w 1985 r

	Wartość rynkowa [mln dol.]	Pozycja na liście w 1984 r.	Wartość sprzedaży [mln dol.]	Przyrost wartości sprzedaży [%]
1 International Business Machines	91697	1	50056	9
25 Hewlett-Packard	10790	19	6571	4
40 Digital Equipment	9137	38	7029	13
61 Xerox	6489	75	8732	4
80 Motorola	5279	67	5443	—2
109 NCR	3940	120	4317	9
112 Tandy	3851	147	3137	13
135 Honeywell	3387	99	6625	9
145 Intel	3163	89	1365	—16
147 Texas Instrument	3121	100	4935	—14
166 Sperry Univac	2870	134	5736	18
169 Burroughs	2853	126	5038	5
176 Wang Laboratories	2721	76	2428	0
197 Warner Comm. Atari	2536	269	2235	10
256 Cray Research	2035	417	3800	66
289 Apple Computer	1721	184	1754	—8
415 Prime Computer	1156	378	770	20
471 Data General	985	240	1199	—4
472 Tandem Computers	978	409	635	12
491 Control Data	934	253	3680	0
592 Amdahl	723	494	862	11
646 LSI Logic	641	727	124	72
683 Varian Associates	593	395	963	0
717 Zenith Electronics	561	641	1624	—5
742 Computer Sciences	529	944	801	13
759 Apollo Computer	512	456	296	37
886 Computer Associates	399	927	170	47
901 Compaq Computer	388	960	504	53
969 Monolithic Memories	337	876	172	—14
982 Stratus Computer	332	955	245	23

Jak więc na tle ogólnego stanu przemysłu komputerowego w USA przedstawia się sytuacja czołowych firm?

DUŻE SYSTEMY KOMPUTEROWE

W IBM, pomimo dochodu ok. 6,5 mld dol. w 1985 r., nastąpiła obniżka notowań akcji.

Od 1970 r. w tej firmie opracowano i wprowadzono na rynek 15 różnych architektur komputerów o nieprzenośnym oprogramowaniu (tab. 2). Pokryto prawie wszystkie potrzeby rynku, jednak przysporzyło to firmie wiele kłopotów.

Tabela 2. Historia osiągnięć IBM w latach 1970-1986

Rok	Opracowanie
1970	System 370 dużych komputerów stanowiących podstawę linii 4300, 308X, 309X
1976	Seria 1 — minikomputer komunikacyjny
1978	Model 8100 — komputer administracyjny
1979	System 5520 — przetwarzanie tekstów
1979	System 38 — minikomputer dla średnich przedsiębiorstw
1980	Autonomiczny system przetwarzania tekstów
1981	Rodzina mikrokomputerów osobistych IBM PC
1983	System 36 — mikrokomputer dla małych przedsiębiorstw i prac administracyjnych
1986	RT/PC — komputer dla naukowców i inżynierów

W 1985 r. ze sprzedaży dużych systemów komputerowych, głównie serii 370, uzyskano prawie 65% z ogólnych zysków. Jednocześnie jednak od 20 do 40% zmniejszono opłaty za użytkowanie starszych systemów. To działanie mające na celu zachęcenie nowych klientów spowodowało, że najnowszy model serii 370 — System 3090 pod nazwą Sierra kosztujący ponad 2 mln dol. — nie znalazł na razie szerszego popytu. Potencjalni użytkownicy twierdzą, że za znacznie niższą cenę mogą użytkować dotychczas posiadany sprzęt typu 308X, a Sierra jest za droga. W kwietniu obniżono więc jej cenę o 10%, a także o 30% obniżono cenę systemu 4030. Również rozczarowaniem dla firmy była sprzedaż minikomputerów serii system 36. System ten bowiem jest już przestarzały technologicznie i niekompatybilny z innymi.

IBM zanotował jednak przyrost dochodów ze sprzedaży oprogramowania o 30% i z serwisu o 16%.

Inni producenci dużych systemów, jak Amdahl Corp., produkujący tańsze komputery odpowiadające architektonicznie systemowi 370 oraz Control Data Corp. i Honeywell Inc., planują obniżenie cen o 15—20% w 1986 r. Jedynie Burroughs Corp., drugi co do wielkości producent dużych komputerów, wierzy w swoją gwiazdę, licząc na sukces nowego komputera A Series, przewidywanego do sprzedaży w 1987 r. Pozostali zaś producenci superkomputerów, jak: Cray, NCR, Sperry (Univac) oraz Prime raczej milczą, licząc zapewne na duże zamówienia wojskowe.

PORZĄDKI W IBM

Nowy prezes IBM John F. Ackers (od czerwca 1985 r.), rozpoczął zmiany w zarządzaniu i zakresie produkcji. Przede wszystkim postanowiono pozostać tylko przy trzech rodzinach komputerów — systemie 370, systemie 36 oraz PC (o tym poniżej). Komputery systemu 36 zwiększą swoją moc w kierunku obsługi większych jednostek finansowych oraz uzyskają wymienność oprogramowania z systemem 370. Z kolei ten ostatni zostanie w najbliższych latach zmniejszony do rozmiaru komputera biurowego, stając się stopniowo odmianą systemu 36. Pozostałe systemy cyfrowe produkowane przez IBM, takie jak 5520 i 8100 otrzymają możliwość wymiany informacji z wymienionymi poniżej, ale nie będą już dalej rozwijane.

Jednocześnie w IBM podjęto działania w kierunku zmniejszenia kosztów ogólnych firmy przez ograniczenie w 1986 r. o 1 mld dol. wydatków na podróże i konferencje. Dąży się też do zmniejszenia, drogą naturalnej redukcji, personelu firmy, liczącego w 1985 r. 406 tys. osób. Upraszczając zarządzanie firmą zreorganizowano też dział sprzedaży zatrudniający 22 tys. osób.

Według przewidywań ekspertów, w 1986 r. IBM uzyska dochód 55 mld dol., a w 1990 r. jego kapitał przekroczy 100 mld dol. Na ostateczne wnioski należy jednak poczekać.

WALKA O RYNEK

HP, drugi — znacznie słabszy kapitałowo — potentat przemysłu komputerowego, obecny stan posiadania zawdzięcza przede wszystkim dochodom ze sprzedaży elektronicznych przyrządów pomiarowych — z czego jest znany na całym świecie. Firma ta w sposób mało przemyślany przestawiała się na produkcję komputerów, produkując w rezultacie trzy różne niekompatybilne ze sobą serie.

Nowy dyrektor HP, John A. Young od 1978 r., pod czujnym okiem Williama R. Hewletta i Davida Packarda, postanowił dokonać przeobrażenia firmy, porządkując również profil produkcji komputerów. Po zatrudnieniu byłego pracownika IBM, Joela Birnbauma (projektanta IBM RT/PC), rozpoczęto prace nad nową serią maszyn HP Spectrum, opartą na architekturze RISC. Nowy komputer HP 3000 serii 930 będzie konkurencyjny w cenie w stosunku do produktów DEC i IBM. Pierwsze egzemplarze tego komputera, w pełni oprogramowane, pojawią się w sprzedaży pod koniec 1986 r. Kolejny, większy komputer tego samego typu, serii 950 nie pojawi się na rynku wcześniej niż pod koniec 1987 r. Pesymiści twierdzą, że architektura RISC nie jest na tyle sprawdzona, aby mogła być stosowana w seryjnych produktach. Na razie udział firmy na rynku minikomputerowym spadł z 4,2% w 1983 r. do 3,9% w 1985 r., podczas gdy dla DEC wzrósł z 5 do 7%. Podobnie na rynku minikomputerów do zastosowań przemysłowych udział ten spadł z 12 do 7,8%. Licząc na sukces, HP traci rynek zbytu. Prezes firmy dokonuje cięć w płacach oraz tworzy specjalne zespoły tylko do spraw reklamy i sprzedaży, gdyż dotychczas przeważało nastawienie inżynierskie na opracowanie i wyprodukowanie — bez troski o promocję sprzedaży.

W CIENIU POTENTATA

DEC jest obecnie najlepiej notowaną na giełdzie firmą produkującą komputery. W ostatnim kwartale 1985 r. firma uzyskała 14% wzrostu sprzedaży oraz wzrost zysków o 20%. Przewiduje się, że w 1986 r. DEC osiągnie zyski przekraczające 550 mln dol. przy 7,7 mld dol. wartości sprzedaży. Rok 1987 powinien firmie przynieść dalszy wzrost zysków o 40% przy 15% wzroście sprzedaży.

Jaka jest przyczyna tego sukcesu firmy produkującej głównie minikomputery? Za podstawową przyczynę uważa się to, że w cieniu IBM, DEC potrafił konsekwentnie rozwijać jedną linię architektury, przechodząc od starej już serii PDP-11 do nowej VAX, zawierającej zarówno stanowiska robocze (ang. workstation) Micro VAX II oraz superminikomputery VAX 750 i VAX 8600. Jednocześnie jednolite oprogramowanie całej linii pozwala użytkownikom na łagodną wymianę systemu cyfrowego na nowszy. Nowe oprogramowanie jest nieco trudniejsze w użytkowaniu dla mniej wprawnych użytkowników, jednak możliwość wykorzystania systemu Unix firmy AT&T zmniejsza tę niedogodność.

Sprzęt produkowany przez DEC jest przenaczony zarówno dla naukowców i inżynierów, jak też do obsługi banków, biur i agencji rządowych. Jednocześnie DEC umiejętnie wykorzystał brak jednolitości w serii komputerów IBM, oferując swoje rozwiązania w celu połączenia różnych typów komputerów w jedną sieć — włącznie z systemem 370 i systemem 36. Do 1985 r. DEC połączył 173 tys. komputerów i wyparł IBM u wielu dotychczas jego stałych użytkowników. Sukces odniosła również realizacja sieci w standardzie Ethernet. Obecnie DEC wprowadza nową serię superminikomputerów VAX 785, zachowując tradycyjną linię oprogramowania. Architektura tej serii, jak twierdzą pracownicy firmy, jest wystarczająco nowoczesna, aby zapewnić sprzedaż na najbliższe 10 lat.

Pomimo tych sukcesów firma DEC ma swoje trudności. Efektywność działania jest w dużym stopniu ograniczona słabą promocją komputerów oraz zupełnym brakiem doświadczenia w prowadzeniu sprzedaży detalicznej i niedowładem organizacyj-

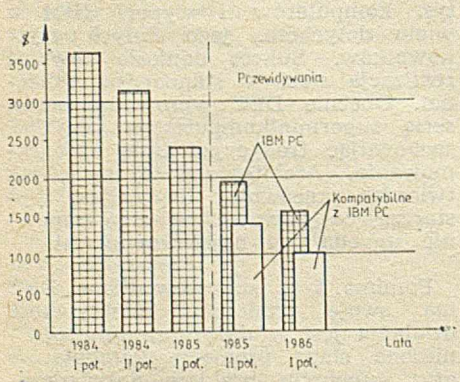
nym. Zysk na jednego zatrudnionego w DEC wynosi 77 tys. dol. w stosunku do 125 tys. dol. w IBM. Koszty rosną szybciej niż zyski. Sredni przyrost zysków od włożonego kapitału akcyjnego jest jednym z najniższych w całym przemysle.

Prezes firmy Kenneth H. Olsen, a szczególnie wiceprezes M. Osterhoff, działają na rzecz zmiany tego stanu, zmieniając strukturę organizacyjną firmy oraz redukując liczbę zatrudnionych z 32 do 27 tys.

Przy okazji warto wspomnieć o konkurentach DEC — firmie Apollo Computer oraz Sun Microsystems produkujących, głównie dla inżynierów, stacje robocze i minikomputery mogące pracować w sieciach. Te komputery są tańsze i szybciej przekształcane na nowe modele. Inny znany producent minikomputerów, Wang Laboratories Inc., przeżywał spore trudności. Tracąc dużo w 1985 r., ma nadzieję osiągnąć zysk w 1986 r., podpisując kontrakt z amerykańskimi firmami (US Air Force) na dostawę minikomputerów za sumę 480 mln dol.

RYNEK MIKROKOMPUTEROWY

Pojawienie się mikrokomputera Apple przeznaczonego do prywatnych zastosowań, w edukacji oraz biurach, zrewolucjonizowało rynek komputerów. Firma Apple Computer Inc. przebojem weszła do grona największych przedsiębiorstw w USA. Z czasem jednak zabrakło jej oddechu, a konkurencja pojawiła się bardzo szybko. Największe przebicie miała oczywiście firma IBM, która postanowiła zdobyć ten rynek dla siebie. Opracowany w 1981 r. mikrokomputer osobisty IBM PC, z odkupionym od Microsoftu oprogramowaniem, stał się wiodący na rynku. Obecnie na świecie wyprodukowano ok. 6 mln mikrokomputerów typu IBM PC. W 1984 r. typowy zestaw kosztował 3,5 tys. dol., obecnie jego cena spadła do 1600 dol., a jego modele innych firm kosztują ok. 15% taniej (rys. 3). Taki spadek cen był możliwy dzięki znacznej obniżce cen elementów. Układ pamięci kosztujący 4 dol. w 1983 r. kosztował 30 centów w 1985 r., a cena napędu dyskietki ze 100 dol. zmalała do 55.



Rys. 3. Kształtowanie się średnich cen mikrokomputerów osobistych

Elementy elektroniczne do IBM PC kosztowały z końcem 1985 r. ok. 400 dol., a jego złożenie w firmie IBM 10—15 dol. od sztuki.

Dla IBM wyrosła potężna kokurencja produkująca taniej, ale nie zawsze lepiej. Firma Tandy Corp. oferowała Tandy 1000, czyli IBM PC/XT za 1000 dol. sprzedając w grudniu 1985 r. 25 tys. sztuk. Inna znana firma Compaq Computer Corp. oferowała mikrokomputer typu IBM PC z monitorem graficznym nie wymagającym dołączenia dodatkowej karty graficznej, zapewniając sobie 15% rynku sprzedaży mikrokomputerów. Podobne osiągnięcia odnotowały: Zenith Electronics, Grid Systems, AT&T i wiele innych. Również HP produkuje mikrokomputery kompatybilne z IBM PC. Jednocześnie rozpoczął się eksport do USA modeli IBM PC z Korei Południowej w liczbie 50 tys. sztuk w 1985 r. oraz z Japonii, z firmy SEIKO, ze sprzedażą 6 tys. sztuk miesięcznie. Niepoślednią rolę odegrał Tajwan, przy czym jego produkty są uważane za najmniej niezawodne.

Tak duża podaż i pewne nasycenie rynku już nieco przestarzałym produktem spowodowało spadek udziału IBM w sprzedaży jego oryginalnego IBM PC. W 1984 r. udział ten wynosił 34%, w rok później już 28%, a w 1986 r. zmalał do 24%. Niewiele też pomogło wprowadzenie kolejnej wersji IBM PC/AT oraz zapowiedź zwiększenia funkcjonalności IBM PC w kierunku jego połączenia i współpracy z większymi komputerami IBM. Przewiduje się, że w 1986 r. IBM będzie zmuszony obniżyć cenę o 25% do 1000 dol.

Wspomniana już firma Apple, po odejściu S. Woźniaka i S. Jobsa, została poddana kuracji odświeżającej przez nowego prezesa Johna Sculleya (dotychczas zajmującego stanowisko prezesa Pepsi-Cola Inc.). Obiecuje on opracowanie nowych typów mikrokomputerów dla edukacji, biur i zastosowań domowych. Pod nowym zarządem przewiduje się osiągnięcie przez Apple dochodów ok. 72 mln dol. w 1985 r. i 140 mln dol. w 1986 r.

Jednak IBM się nie poddał. Obecnie oferowany jest nowy jakościowo — przenośny i składany (ang. portable convertible) — IBM PC ważący ok. 5,5 kg. Należy on do nowej rodziny mikrokomputerów — proszę się nie śmiać — leżących na łonie osoby siedzącej (ang. laptop) i jest przeznaczony do pracy zarowno w podróży, jak i w domu. Mikrokomputer ten kosztuje 1995 dol. i jest wyposażony w nowy typ 3-calowej dyskietki (co niektórzy uważają za jego poważną wadę przy akceptacji przez dotychczasowych użytkowników dyskietek 5¼-calowych). Mikrokomputer ten jest wyposażony w płaski ekran, a za dodatkowe 505 dol. może być podłączony do niego typowy monitor.

Przewiduje się sukces IBM na rynku, gdyż konkurencja nie dysponuje jeszcze produktem tak nowym jakosciowo. Dostępne są jedynie przenośne mikrokomputery bateryjne. Sprzedaż „laptopów" (ogłaszamy konkurs na polską nazwę) w 1986 r. przewidywana jest na sumę 967 mln dol., co jednak jest kroplą w morzu całego rynku mikrokomputerów ocenianego na 14,5 mld dol.

CZY IBM JEST LIDEREM NA RYNKU KOMPUTEROWYM?

Można teraz pokusić się o udzielenie odpowiedzi na postawione w tytule pytanie. Oczywiście wydaje się, że odpowiedź powinna być pozytywna. Ułomności w profilu i zakresie produkcji w IBM, dość irytujące obecnie, zostaną chyba wkrótce przezwyciężone. Trzeba tutaj wspomnieć, że IBM w 1986 r. przeznaczył na badania 4,7 mld dol., a na automatyzację swoich zakładów 3 mld dol., co daje sumę o 1 mld dol. przekraczającą całą wartość sprzedaży DEC.

Zakłada się też, że obecna stagnacja na rynku komputerów ma charakter przejściowy i już w następnych latach nastąpi wzrost popytu na nowe komputery każdej wielkości. Ostatnie dwa lata były okresem mobilizującym wiele firm do uporządkowania swojego własnego podwórka przed nowymi podbojami informatyki już wszystkich dziedzin życia. Twierdzi się, że w USA już w 1987 r. każdy pracownik zatrudniony w jakiejkolwiek firmie, od sekretarki wzwyż, bez znajomości użytkowania komputera stanie się zbędny.

Oprac. WACŁAW ISZKOWSKI
na podstawie BUSINESS WEEK
z marca i kwietnia 1986 r.

EGZEMPLARZE ARCHIWALNE czasopisma można nabywać w Klubie Prasy Technicznej w Warszawie, ul. Mazowiecka 12, tel. 27-43-65 lub zamówić w Dziale Handlowym Wydawnictwa NOT SIGMA, ul. Bartycka 20, skr. poczt. 1004, 00-950 Warszawa, tel. 40-37-31, na rachunek dla instytucji lub za zaliczeniem pocztowym dla osób fizycznych.

Terminologia lokalnych sieci komputerowych(4)

W obecnym, ostatnim odcinku rozważań terminologicznych dotyczącym słownictwa lokalnych sieci komputerowych, przedstawiono elementy struktury tych sieci. Głównym pojęciem z tej dziedziny jest omówiona w numerze 11/1986 **konfiguracja** sieci, w piśmiennictwie polskim nazywana także **topologią** (od ang. topology). W dużych sieciach komputerowych konfiguracją nazywa się rozmieszczenie węzłów i łączy. Natomiast według cytowanej wielokrotnie w »Informatyce« normy terminologicznej IEEE Std 739, konfiguracją nazywa się, bardziej ogólnie, ułożenie systemu komputerowego lub sieci komputerowej, określone przez rodzaj, liczbę i główne właściwości jednostek funkcjonalnych (the arrangement of a computer system or network as defined by the nature, number and the chief characteristics of functional units).

Główne rodzaje konfiguracji omówiono w numerze 1/1987, gdzie wspomniano o podawanym niekiedy subtelnym rozróżnieniu między pętlą a pierścieniem. Według [5] podstawową cechą **pętli** jest istnienie pojedynczej stacji centralnej, dołączonej do obu jej końców, sterującej indywidualnym dostępem do ośrodka mającego charakter zamkniętego obwodu. Wszystkie inne urządzenia mogą kontaktować się wzajemnie dopiero po wydaniu zezwolenia na dostęp do ośrodka. **Pierścień** różni się od pętli tym, że nie ma centralnej stacji sterującej i wszystkie stacje dołączone są na równych prawach. Stacje w pierścieniu są połączone w ciąg o strukturze zamkniętej, tzn. każda stacja łączy się z dwoma sąsiednimi.

W wielu sieciach lokalnych, np. w sieci Ethernet, stacja łączy się z ośrodkiem przez sterownik, kabel odgałęźny (ang. drop cable), nadajnik-odbiornik i odczep zwany również rozgałęźnikiem. Przez **odczep** (ang. tap) rozumie się złącze służące do mechanicznego i elektrycznego połączenia ośrodka transmisji z kablem odgałęźnym.

Do najważniejszych rodzajów stacji należą tzw. **stacje obsługi** (ang. server), tj. indywidualne urządzenia zapewniające innym urządzeniom dołączonym do sieci określony rodzaj usług, np. obsługę plików, obsługę wydruków, obsługę terminali itp. Ważnym rodzajem stacji jest tzw. **koncentrator**, tj. urządzenie służące do pełnienia funkcji komunikacyjnych dla dużej liczby innych urządzeń o małej szybkości, przez jeden lub kilka kanałów o dużej szybkości. Trzeba jednak przyznać, że jest to technika stosowana głównie w dużych sieciach komputerowych. W tych sieciach występują również inne duże urządzenia, jak: komputer obliczeniowy (ang. host), procesor wysunięty (ang. front-end processor) czy węzeł (ang. node).

Coraz częściej w sieciach lokalnych spotyka się wydzielone fizycznie grupy terminali, stacji lub innych urządzeń, wyróżnione jako logiczna całość ze względu na określone kryterium. Taką grupę określa się po angielsku mianem cluster. Spośród licznych polskich odpowiedników terminu angielskiego, z jakimi się zetknąłem, można wymienić następujące: gromada, skupienie, grono i wiązka. Ze względu na brak jednolitej opinii co do uzgodnienia odpowiednika polskiego, zaproponowałbym spolszczenie terminu angielskiego i nazwanie tej grupy urządzeń wyrazem **klaster**.

W ten sam sposób proponuję spolszczyć nazwę innego ważnego elementu rozbudowy sieci, jakim jest tzw. **repeter** (ang. repeater). Repeter jest urządzeniem służącym do zwiększenia długości ośrodka fizycznego lub rozszerzenia konfiguracji powyżej ograniczeń pojedynczego segmentu, do maksymalnie dopuszczalnej długości dwupunktowej linii transmisyjnej. Repetery wykonują podstawowe funkcje odtwarzania amplitudy, kształtu i przebiegu czasowego sygnału w odniesieniu do wszystkich sygnałów danych i kolizji.

W łączeniu sieci lokalnych między sobą odróżnia się przypadki sieci podobnych i sieci odmiennych. Urządzenie łączące dwie lokalne sieci komputerowe o tym samym protokole warstwy sieciowej nazywa się **pomostem** (ang. bridge). Natomiast urządzenie łączące sieci komputerowe o różnych protokołach nazywa się **bramą** (ang. gateway), choć spotkałem też inne polskie nazwy, jak np. furtka. Brama stanowi sprzężenie między protokołami wyższych warstw i jest bardziej złożona od pomostu.

Wśród właściwości sieci lokalnych związanych z ich konfiguracją szczególnie ważną rolę odgrywają dwie, określone po angielsku jako cohesion i connectivity. Pierwszą z nich można nazwać po polsku **spoistością** (nie — spójnością, gdyż ten termin ma inny odpowiednik angielski) i określić jako minimalną liczbę połączeń, których usunięcie powoduje podział sieci przynajmniej na dwie rozłączne części. Drugą właściwość, polegającą na zdolności sieci do tworzenia połączeń z dużą liczbą różnych systemów, można nazwać po polsku **łączliwością**, choć oczywiście jest to neologizm.

Kończąc omawianie terminologii tej nowoczesnej dziedziny techniki komputerowej należy podkreślić, że wiele przedstawionych tu terminów jest jedynie propozycją nazewnictwa. Większość podanych określeń i definicji stanowi rozwinięcie lub uzupełnienie znanych określeń pochodzących z innych opracowań [1—8]. W poszukiwaniu polskich odpowiedników dla wielu terminów angielskich szczególnie przydatna okazała się bardzo dobrze przetłumaczona książka [1].

LITERATURA

[1] Davies D. W., Barber D. L. A.: Sieci teleinformatyczne. WNT, Warszawa, 1979
[2] ECMA: Local Area Networks — CSMA/CD Baseband — Link Layer. Standard ECMA-82, European Computer Manufacturers Association, Geneva, 1984
[3] ECMA: Local Area Networks — Token Bus Technique. Standard ECMA-90, European Computer Manufacturers Association, Geneva, 1983
[4] IEEE: Local Area Networks — CSMA/CD Access Method and Physical Layer Specifications. IEEE Std. 802.3-1985, Institute of Electrical and Electronics Engineers, New York, 1985
[5] Gandy M.: Microcomputer Communications. NCC Publications, Manchester, 1985
[6] ISO: Data Processing Vocabulary — Section 09 — Data Communications. Standard ISO 2382/IX, International Organization for Standardization, Geneva, 1977
[7] Maroński J., Rupińska M.: Computer networks — terminology. PWN, Warszawa, 1980
[8] Rupińska M.: Sieci komputerowe — słownictwo. Prace IPI PAN, nr 350. Instytut Podstaw Informatyki PAN, Warszawa, 1979.

JANUSZ ZALEWSKI

Kowalski R.: Programowanie logiczne i komputery nowej generacji (1)
INFORMATYKA 1987, nr 3, s. 2
Charakterystyka programowania logicznego oraz jego wpływu na rozwój technologii programowania i zastosowań.

Kowalski R.: Logical programming and new generation of computers (1)
INFORMATYKA 1987, No. 3, p. 2
Characteristics of logical programming and its influence on development of programming technology and applications.

Kowalski R.: Logische Programmierung und Rechner der neuen Generation (1)
INFORMATYKA 1987, Nr. 3, S. 2
Eine Charakteristik der logischen Programmierung und ihres Einflusses auf die Entwicklung der Programmierungstechnologie und der Anwendungen.

Łukaszewicz W.: Przechowywanie wiedzy i niewiedzy w bazach danych (2)
INFORMATYKA 1987, nr 3, s. 5
Druga część artykułu na temat baz wiedzy, zawierająca omówienie formalnych metod wnioskowania niemonotonicznego.

Łukaszewicz W.: Knowledge and ignorance storaging in data bases (2)
INFORMATYKA 1987, No. 3, p. 5
Second part of the paper on knowledge data bases, which include discussion of formal methods for non-monotonic inference.

Łukaszewicz W.: Wissenschafts- und Ignoranzspeicherung in Datenbanken (2)
INFORMATYKA 1987, Nr. 3, S. 5
Zweiter Teil des Artikels zur Thema der Wissenschaftsbanken, der eine Besprechung von formellen Methoden der nichtmonothonischen Folgerung.

Cofta P.: Czy następuje zmierzch języków asemblerowych?
INFORMATYKA 1987, nr 3, s. 8
Historia, stan obecny oraz perspektywy rozwoju języków asemblerowych.

Cofta P.: Is it a recession of assembly languages?
INFORMATYKA 1987, No. 3, p. 8
History, actual situation and development perspectives of assembly languages.

Cofta P.: Ist schon der Untergang von Assemblersprachen?
INFORMATYKA 1987, Nr. 3, S. 8
Geschichte, heutiger Stand und Entwicklungsaussichten der Assemblersprachen.

Grzywak A., Kołodziej J.: Sieci komputerowe oparte na mikrokomputerach MERA 60
INFORMATYKA 1987, nr 3, s. 11
Charakterystyka sieci mikrokomputerowej opartej na sprzęcie krajowym i zrealizowanej w dziedzinie energetyki.

Grzywak A., Kołodziej J.: Networks based on Mera 60 microcomputers
INFORMATYKA 1987, No. 3, p. 11
Characteristics of the microcomputer network, based on home hardware and realized in power industry.

Grzywak A., Kołodziej J.: Rechnernetz mit Anwendung von Mera 60-Mikrorechnern
INFORMATYKA 1987, Nr. 3, S. 11
Eine Charakteristik des Mikrorechnernetzes, der mit einheimischer Hardware im Bereich der Energetik realisiert wurde.

Góral S., Janczewski K., Janyszek J., Zajączkowski Z.: Użytkowanie Międzyuczelnianej Sieci Komputerowej
INFORMATYKA 1987, nr 3, s. 15
Charakterystyka zasad eksploatacji oraz uruchomienia i wdrożenia sieci komputerowej zrealizowanej na Politechnice Wrocławskiej.

Góral S., Janczewski K., Janyszek J., Zajączkowski Z.: Use of th Interuniversity Computer Network
INFORMATYKA 1987, No. 3, p. 15
Characteristics of exploitation principles and initiating of the Interuniversity Computer Network, realized in the Wrocław Technical University.

Góral S., Janczewski K., Janyszek J., Zajączkowski Z.: Nutzung des Hochschulrechnernetzes
INFORMATYKA 1987, Nr. 3, S. 15
Eine Charkteristik der Betriebsgundsätze und des Einsatzes von Hochschulrechnernetz, der an der Technischen Universität in Wrocław realisiert wurde.

Trawiński G.: Pakiet zintegrowany Computex
INFORMATYKA 1987, nr 3, s. 19
Charakterystyka uniwersalnego wielofunkcyjnego pakietu programowego przeznaczonego do eksploatacji na mikrokomputerach IBM PC i kompatybilnych.

Trawiński G.: Computex integrated package
INFORMATYKA 1987, No. 3, p. 19
Characteristics of the universal multifunctional program package, which is appropriated for applications on IBM PC and compatible microcomputers.

Trawiński G.: Computex — ein integriertes Programmpaket
INFORMATYKA 1987, Nr. 3, S. 19
Eine Charakteristik des universellen, multifunktionellen Programmpakets, das für Anwendung mit IBM PC und kompatiblen Mikrorechner bestimmt ist.

Mokrzycki W.: System programowania grafiki komputerów PSG 2.2.
INFORMATYKA 1987, nr 3, s. 23
Charakterystyka uniwersalnego systemu programowania grafiki komputerów, opracowanego w Instytucie Maszyn Matematycznych w Warszawie.

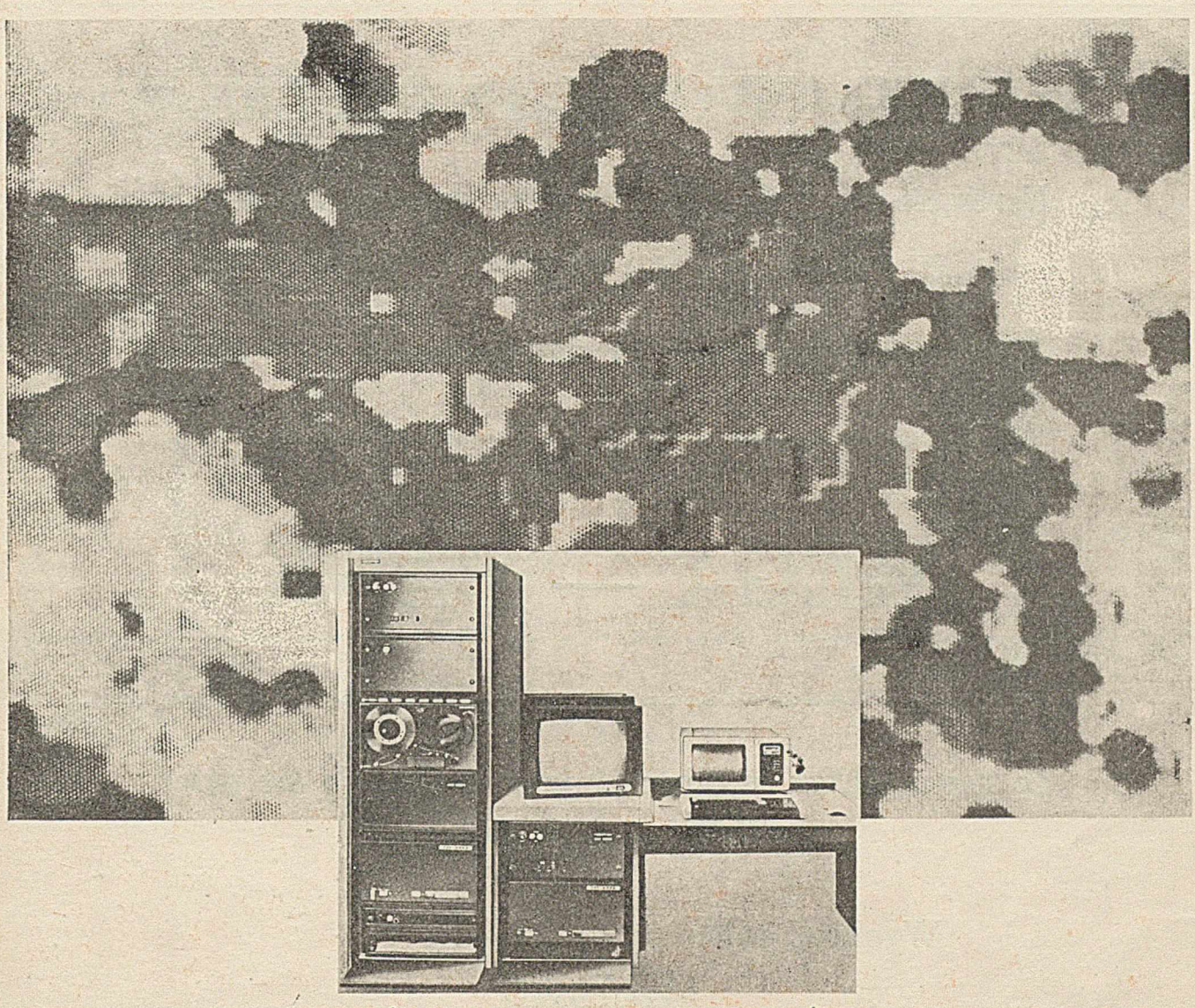
Mokrzycki W.: PSG 2.2. — ein Programmiersystem für Computergrafik
INFORMATYKA 1987, Nr 3, S. 23
Eine Charakteristik des universellen Systems für Comptuergrafikprogrammierung, das im Institut für Mathematische Maschinen in Warschau erarbeitet wurde.

Mokrzycki W.: PSG 2.2. — a system for computer graphics
INFORMATYKA 1987, No. 3, p. 23
Characteristics of the universal system for computer graphics programming, which was elaborated in the Institute of Computers in Warsaw.

Rokita P.: Nowa wersja programu szeregującego wysokiego poziomu dla systemu operacyjnego George-3
INFORMATYKA 1987, nr 3, s. 25
Ulepszenie programu szeregującego dla systemu operacyjnego George-3, zrealizowane na Politechnice Warszawskiej.

Rokita P.: A new version of a high level arranging program for George-3 operating system
INFORMATYKA 1987, No. 3, p. 25
Improvement of the arranging program for George-3 operating system, realised at the Warsaw Technical University.

Rokita P.: Eine neue Version des Gliederungsprogramms von hohen Niveau für George 3-Betriebssystem
INFORMATYKA 1987, Nr. 3, S. 25
Eine Vervollkommnung des Gliederungsprogramms für George 3-Betriebssystem, die an der Technischen Universität in Warszawa realisiert wurde.



Pakiet programowy do sporządzania map tematycznych

A 6471-73

Za pomocą pakietu programowego „Thematische Kartierung" (Tematyczne Sporządzanie Map) firmy Robotron oraz cyfrowego systemu przetwarzania obrazów robotron BVS A 6470 można interakcyjnie przetwarzać wielospektralne obrazy powierzchni Ziemi, przekazywane z takich satelitów jak LANDSat, Spot, Meteor czy Fragment, a także z samolotów.

Pakiet ten, łącznie z oprogramowaniem podstawowym systemu BVS, pozwala realizować następujące procedury:

- badanie obrazów wielospektralnych pod względem ich treści, stanu oraz wyboru scen cząstkowych,
- poprawianie jakości obrazu za pomocą zwiększania konstrastowości i metody filtrowania,
- analiza wielospektralna,
- geometryczna korekcja i rejestracja, dostosowująca obrazy satelitarne do siatki kartograficznej,
- wyprowadzanie skorygowanych obrazów w postaci filmu lub wydruku.

System robotron BVS A 6470 przewarza dane obrazowe zgodnie z normą zapisu video, a najbardziej rozbudowany model BVS A 6473 został dostosowany do potrzeb ośrodków operatywnego przetwarzania obrazów

Robotron Export-Import
Allee der Kosmonauten 24
Berlin DDR – 1140

Zapraszamy do odwiedzenia naszej ekspozycji na wystawie Infosystem '87, w dniach 7 – 11 kwietnia 1987 r., Wrocław, Hala Ludowa, ul. Wystawowa.

robotron

EO/1435/K/86

KONIEC TWOICH PROBLEMÓW

Nareszcie wszystkie potrzebne Ci dane dotrą na czas
w przejrzystej formie tabel i wykresów.

Pakiet oprogramowania na mikrokomputery 16-bitowe

MEGA — BANK

to nowe MEGA możliwości jakie otwierają się przed
Twoim przedsiębiorstwem.

Wyobraź sobie

MILIARD

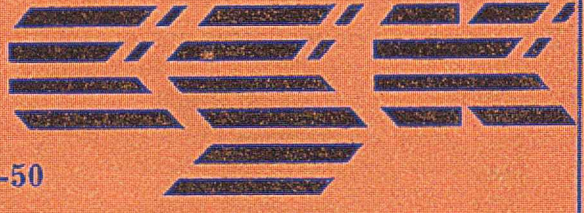
rekordów, które możesz
zapełnić według własnych potrzeb i uznania.

Miliard kooperantów, miliard pracowników, miliard produktów
— z tym wszystkim nasz MEGA-BANK poradzi sobie bez trudu.

System jest łatwy w obsłudze i opracowany w języku polskim.

Gwarantujemy satysfakcję!

COMPUTER STUDIO KAJKOWSCY
PROFESJONALNE OPROGRAMOWANIE MIKROKOMPUTERÓW
ul. Balladyny 3B, 81-524 Gdynia, tel.: 29-00-18, 24-01-50



EO174/K/87